



Defence Research and  
Development Canada

Recherche et développement  
pour la défense Canada



# **Local Navigation for Unmanned Ground Vehicles**

*A Survey*

J. Giesbrecht  
DRDC Suffield

Technical Memorandum  
DRDC Suffield TM 2005-038  
December 2005

Canada



# **Local Navigation for Unmanned Ground Vehicles**

*A Survey*

J. Giesbrecht  
Defence R&D Canada – Suffield

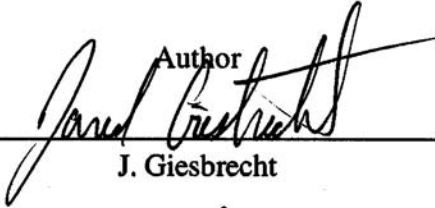
**Defence R&D Canada – Suffield**

Technical Memorandum

DRDC Suffield TM 2005-038

December 2005

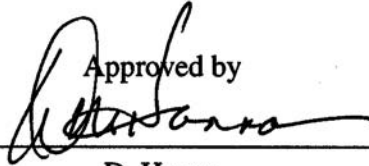
Author



---

J. Giesbrecht

Approved by

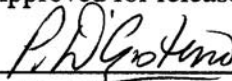


---

D. Hanna

Head/Tactical Vehicle Systems Section

Approved for release by



---

Dr. P. D'Agostino

Chairman/Document Review Panel

## **Abstract**

---

This document surveys existing methods of local navigation of mobile robots. It analyzes the state of the art in both indoor obstacle avoidance techniques and those for outdoor rough terrain. A variety of techniques such as Potential Fields, Vector Field Histogram, and Dynamic Window are examined in detail, in addition to outdoor systems such as Ranger, Morphin and the NIST Demo III architecture. Finally, it discusses the applicability of robot motion planning algorithms like Rapidly Exploring Random Trees to the robot navigation problem.

## **Résumé**

---

Ce document évalue les méthodes actuelles de navigation locale des robots mobiles. Il analyse l'état actuel des techniques d'évitement d'obstacles d'intérieur et de mauvais terrains extérieurs. Une variété de techniques telles que Potential Fields, Vector Field Histogram, et Dynamic Window sont examinées en détails en plus des systèmes extérieurs tels que Ranger, Morphin et l'architecture NIST Demo III. Il discute enfin de l'applicabilité des algorithmes de planification de la motion des robots, tels que Rapidly Exploring Random Trees, au problème de la navigation des robots.

This page intentionally left blank.

## Executive summary

---

**Background:** Intelligent Unmanned Ground Vehicles (UGVs) must automatically find a safe and efficient way from point A to B in the face of obstacles and unknown terrain. Local navigation techniques, the focus of this survey, can be classified as either indoor or outdoor. Indoor systems model discrete obstacles, with vehicle dynamics and stability of lesser concern, and are effective at dealing with rapidly changing worlds and moving obstacles. Outdoor techniques operate on rough terrain at higher vehicle speeds. This document is a survey of all of these types of systems.

**Principle Results:** Indoor obstacle avoidance techniques are well developed, with several effective methods available. Rough terrain navigation is a much more difficult and unsolved problem. Some systems assume that the global path planner has sufficient knowledge, and simply follow its pre-determined path. Other systems use feed-forward control to evaluate robot paths over a digital elevation map, accounting for steering latency, tip-over, body collision and step hazards. Extremely detailed models of the robot and the terrain for physics-based simulation may be used. The application of fuzzy logic, using human-style terrain categorization and judgements on roughness and steepness to determine the best robot motion has also been tried. A final group uses heuristic, probabilistic algorithms to explore the high dimension configuration space to precisely plan the robot's sequence of motions.

**Significance of Results:** Indoor two dimensional obstacle avoidance is more or less a solved problem. A number of systems have shown to be robust under varying indoor conditions and density of obstacles, even navigating among crowds of people. Many researchers working with indoor robots have turned to the problems of localization and map making.

Conversely, much research is still required for outdoor navigation. Although a few systems have shown some competence, robustness to a variety of conditions has not been demonstrated. High speed UGVs have not yet been successful in a variety of environments and sensing conditions.

**FutureWork:** Future systems will begin to incorporate different sensing methods to provide better world models for local navigators (laser range finding, stereo vision, radar and video based image analysis). Increases in processing power may allow the development of one all-encompassing navigation algorithm, but more likely, autonomous systems will switch between algorithms that are optimized for different environments (moving obstacles, rough terrain, etc.). Finally, strategic concerns and operation in complex urban and three dimensional worlds, crucial to today's battlefield, must be investigated.

J. Giesbrecht. 2005. Local Navigation for Unmanned Ground Vehicles. DRDC Suffield TM 2005-038. Defence R&D Canada – Suffield.

## Sommaire

---

**Contexte :** Les véhicules terrestres intelligents sans pilote doivent automatiquement trouver un moyen de se déplacer d'un point A à un point B, parmi des obstacles et sur un terrain inconnu, d'une manière sécuritaire et efficace. Les techniques de navigation locale sur lesquelles on se concentre dans cette évaluation, peuvent être classifiées comme des techniques d'intérieur ou d'extérieur. Les obstacles discrets des modèles de systèmes d'intérieur, pour lesquels le problème de la stabilité et de la dynamique des véhicules est moins important, sont efficaces à gérer les milieux changeants et les obstacles en mouvement. Les techniques d'extérieur opèrent sur des mauvais terrains à des vitesses plus rapides de véhicules. Ce document évalue tous ces types de systèmes.

**Résultats principaux :** Les techniques d'évitement d'obstacles d'intérieur sont bien développées et ont à leur actif plusieurs méthodes efficaces disponibles. Les mauvais terrains de navigation sont un problème beaucoup plus difficile qui n'a pas été encore résolu. Certains systèmes considèrent que le planificateur de parcours global possède une connaissance suffisante et suit tout simplement un parcours prédéterminé. D'autres systèmes utilisent un contrôle d'information dirigée vers l'avant pour évaluer le parcours du robot sur une carte numérique d'altitude en tenant compte du temps d'attente dans la direction, du basculage, des collisions et des dangers d'empiètement. On est en mesure d'utiliser des modèles extrêmement détaillés de robot et de terrain pour la simulation basée sur la physique. On a aussi essayé d'appliquer la logique de l'incertain, en utilisant, à la manière des humains, des catégorisations de terrains et des jugements sur les aspérités et la raideur du terrain pour déterminer les meilleurs mouvements du robot. Un dernier groupe utilise les algorithmes heuristiques de probabilité pour explorer les espaces d'arrangements de grande dimension et planifier précisément les séquences des mouvements du robot.

**La portée des résultats :** L'évitement des obstacles à deux dimensions est plus ou moins un problème résolu à l'intérieur. Un certain nombre de systèmes ont fait preuve de robustesse dans une variété de conditions d'intérieur et de densité des obstacles donc naviguer parmi une foule de personnes. Beaucoup de chercheurs travaillant pour les robots d'intérieur se sont tournés vers les problèmes de localisation et de création de cartes.

Par contre, il faudra continuer les recherches dans le domaine de la navigation d'extérieur. Bien que quelques systèmes aient fait leurs preuves, on n'a pas encore démontré leur robustesse dans des conditions variées. Les véhicules sans pilote de grande vitesse ne fonctionnent pas encore très bien dans une variété de milieux et de conditions de détection.

**Les travaux futurs :** Les systèmes futurs commenceront à incorporer des méthodes différentes de détection pour procurer de meilleurs modèles aux navigateurs locaux (télémétrie à laser, stéréovision, radar et analyses d'images vidéos). L'augmentation de la puissance de traitement pourra permettre le développement d'algorithmes universels de navigation mais il est plus probable que les systèmes autonomes permuteront entre les algorithmes qui auront été optimisés pour chaque contexte différent (obstacles en mouvement, mauvais terrains, etc.) Il faudra enfin étudier les problèmes stratégiques et opérationnels qui s'avèrent cruciaux dans les milieux urbains complexes et tridimensionnels que sont les champs de bataille actuels.



# Table of contents

---

Abstract . . . . .	i
Resume . . . . .	i
Executive Summary . . . . .	iii
Sommaire . . . . .	iv
Table of contents . . . . .	v
List of figures . . . . .	vii
List of tables . . . . .	viii
1. Introduction . . . . .	1
1.1 Background . . . . .	1
1.2 Overview . . . . .	3
2. Indoor Obstacle Avoidance . . . . .	4
2.1 Directional Methods . . . . .	4
2.1.1 Potential Fields . . . . .	4
2.1.2 Vector Field Histogram . . . . .	6
2.1.3 Nearness Diagram . . . . .	8
2.1.4 Ego-Kinematic and Ego-Dynamic Space . . . . .	9
2.2 Curvature Methods . . . . .	9
2.2.1 The Steering Angle Field Approach . . . . .	10
2.2.2 Dynamic Window Approach . . . . .	10
2.2.3 Curvature-Velocity Method . . . . .	11
2.3 Clothoid and Polynomial Methods . . . . .	13
2.3.1 DEMO II and DEMO III . . . . .	14
2.4 Obstacle Avoidance with Moving Obstacles . . . . .	16
2.4.1 Velocity Obstacle Approach . . . . .	16

2.4.2	Dynamic Motion Planning Using Potential Fields . . . . .	18
3.	Rough Terrain Navigation . . . . .	19
3.1	Pure Pursuit Path Tracking . . . . .	19
3.2	Local Navigation Using Terrain Evaluation . . . . .	20
3.2.1	RANGER . . . . .	20
3.2.2	Morphin/Gestalt . . . . .	24
3.2.3	High Speed Hazard Avoidance . . . . .	26
3.3	Physics-Based Vehicle Modeling for Local Navigation . . . . .	27
3.3.1	Physics Based Path Evaluation . . . . .	28
3.3.2	Physical Modeling with Dynamic and Contact Constraints . . .	29
3.4	Fuzzy Logic and Neural Networks . . . . .	31
3.5	Robot Motion Planning . . . . .	35
3.5.1	Configuration and State Space . . . . .	35
3.5.2	Holonomic and Non-holonomic Constraints . . . . .	35
3.5.3	Probabilistic Planning . . . . .	36
3.5.4	Rapidly Exploring Random Trees . . . . .	37
3.5.5	Randomized Motion Planning on Rough Terrain . . . . .	39
4.	Conclusions and Future Work . . . . .	41
	References . . . . .	44

## List of figures

---

Figure 1. A rolled over Sandstorm during testing for the DARPA Grand Challenge. . . .	2
Figure 2. An example Potential Field. . . . .	4
Figure 3. Vector Field Histogram. . . . .	7
Figure 4. Nearness Diagram. . . . .	8
Figure 5. The Dynamic Window Approach. . . . .	11
Figure 6. Curvature-Velocity Method. . . . .	13
Figure 7. Lane-Curvature Method. . . . .	13
Figure 8. Possible problems with arc-based navigation. . . . .	14
Figure 9. Paths evaluated by Demo III obstacle avoidance. . . . .	15
Figure 10. Creating a Velocity Obstacle. . . . .	17
Figure 11. Pure Pursuit Path Tracking. . . . .	20
Figure 12. Inverse and forward vehicle models. . . . .	22
Figure 13. Feed-forward model in Ranger. . . . .	23
Figure 14. Hazard assessment in Ranger. . . . .	24
Figure 15. Arc evaluation in Morphin. . . . .	25
Figure 16. Trajectory space for high speed hazard avoidance. . . . .	27
Figure 17. Curvatures eliminated for obstacles. . . . .	27
Figure 18. Velocities eliminated for negative obstacles. . . . .	27
Figure 19. Forces used in Shiller's rover navigator. . . . .	28
Figure 20. Shiller's velocity limit curves. . . . .	29
Figure 21. Physical models used by Cherif. . . . .	30
Figure 22. Iterative planning with physics-based modeling. . . . .	30
Figure 23. Terrain roughness fuzzy sets. . . . .	33

Figure 24. Terrain sectors for regional traverse. . . . .	34
Figure 25. Fuzzy logic system structure. . . . .	34
Figure 26. A simplified probabilistic roadmap planner. . . . .	37
Figure 27. A simplified Rapidly Expanding Random Tree. . . . .	38

## **List of tables**

---

Table 1. Decision making in the Nearness Diagram method. . . . .	9
Table 2. Weight rules for fuzzy logic system. . . . .	34
Table 3. Comparison of obstacle avoidance techniques. . . . .	42
Table 4. Comparison of outdoor rough terrain techniques. . . . .	43

# 1. Introduction

---

## 1.1 Background

It is easy to underestimate the competency with which humans drive vehicles in outdoor environments (well, most of us anyway). We can effortlessly drive an off road vehicle through woods and fields, down roads and paths, in summer and winter. We perceive, comprehend and make decisions quickly through a wide variety of environments. Most importantly, we can simultaneously avoid obstacles while looking ahead to plan paths through difficult terrain to pursue the destination. We can easily anticipate what actions will make our vehicle unstable, using an intuitive model of how the machine behaves under a wide variety of dynamic conditions and interacts with the terrain. This process is much more difficult for robotic vehicles. In addition to the limitations of sensing equipment, it has been difficult to create software with the sort of object recognition and fast decision making humans accomplish without being conscious of the complexities involved.

The navigation problem for unmanned ground vehicles can generally be divided into two complementary parts. The first, global path planning[1], is the deliberative process of looking ahead through a high level view of the robots world, finding the shortest path to a long-term goal, avoiding cul-de-sacs and undesirable terrain by planning using information the robot has been given. A more reactive process, local navigation, utilizes local sensor data, rather than a global map, searching the immediate environment for hazards and avoiding them while simultaneously seeking a goal. Local navigation handles moving obstacles, incomplete knowledge, vehicle stability and safety, and sensors with limited range. Local navigators generally do not retain global knowledge, or retain acquired information, instead reacting to changing conditions. The maximum safe speed attainable by a UGV is directly related to how fast the local navigator can operate.

Three different fields of robotics are merging in local navigation techniques. The first field comprises methods used by indoor robots in their flat, two dimensional environments to dodge discrete obstacles. The second, physics based modelling and simulation, allows a controller to estimate safe vehicle motions over rough and rolling terrain. The third is motion planning for robotic manipulators.

The merging of these fields has allowed a more cognitive approach to planning the robot's motion over the terrain, which nevertheless operates quickly. However, much progress is still required before these systems will be competent at high speeds over rough terrain. Because real robots have inertia, limited controllability, limited sensing, and operate in dynamic, changing environments, practical systems must:

- Provide safe travel with noisy sensors and dead reckoning errors.
- Be computationally efficient, operate in real time, and be very reactive to sensor measurements.

- Understand the kinematics of the vehicle, and plan within the achievable limits of the vehicle.
- Understand the dynamics of the vehicle, especially at higher vehicle speeds, accounting for inertia and steering response.
- Comprehend the dynamics of the vehicle/terrain interactions and understand the implications of soft soil, small rocks, and slope in order to prevent vehicle instability.
- Be goal directed, explicitly maximizing forward progress.
- Handle incomplete information.

The speed of the local navigator directly limits the speed at which the vehicle can safely operate. At high speeds, the UGV lacks time to sense and plan, given limited sensor technology. Detailed physics based analysis and planning sacrifices speed of execution, limiting maximum vehicle speed. Conversely, if a system operates fast enough to avoid obstacles, but without enough fidelity to the terrain or vehicle, dangerous instability may result (see Figure 1).



**Figure 1:** A rolled over Sandstorm during testing for the DARPA Grand Challenge.

Therefore, it is imperative to provide the most bang for the computational buck, designing the system with only as much fidelity as required, using an optimal combination of the robustness of reactive obstacle avoidance methods with the efficiency and exact control of motion planning algorithms. The results from 2004 DARPA Grand Challenge, in which competitors were offered a \$1,000,000 prize if their UGV could navigate 241km across rough desert terrain, indicate that the state of the art is not yet acceptable. Even in the low complexity environment provided, the furthest any team managed, the aforementioned Sandstorm, was 12km, hampered by the incredibly fast execution time required, demands on vehicle dynamics, and a lack of current sensing technology.

In order to get a complete picture of the UGV navigation problem, it is suggested that the reader also read the survey paper “Global Path Planning for Unmanned Ground Vehicles”[1].

## **1.2 Overview**

The first section of this survey covers obstacle avoidance methods used by indoor mobile robots operating on flat surfaces with discrete obstacles, considering travel in either straight lines, circular arcs, or more complex trajectories called clothoids. In addition, some systems explicitly detect and avoid moving obstacles. The second section reviews rough terrain methods. The most basic use a local feedback controller to follow the global path. More complex terrain evaluation methods rate the terrain traversability and choose the most benign. The physics-based modeling approaches use a detailed model of the robot and the terrain to simulate vehicle/terrain interactions. The fuzzy logic approach uses video based perception and simple rules to determine robot motion. Finally, robot motion planners carefully search the space of all the robot’s degrees of freedom to deliberately plan control. Each method has strengths, weaknesses, and appropriate application domains. The usefulness of each in the control of Unmanned Ground Vehicles will be analyzed as they are surveyed.

## 2. Indoor Obstacle Avoidance

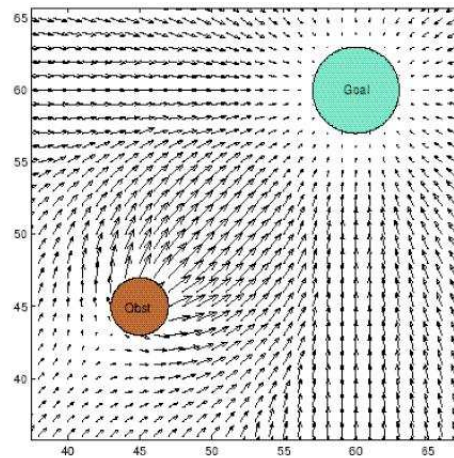
---

Obstacle avoidance algorithms use binary obstacle representations for small robots operating inside offices, reacting quickly to changes, such as moving people. They are simple, iterate quickly and issue new commands at a very high frequency. The first, quite primitive methods, the BUG algorithms[2], simply follow obstacle edges using contact sensors until the robot is able to continue on towards the goal. More recent and complex systems create obstacle maps and include robot kinematics when evaluating actions. Given the flat surfaces, wider robot bases, and lower speeds for indoor vehicles, obstacle avoidance methods assume a two dimensional world with limited concern for vehicle stability and dynamics.

### 2.1 Directional Methods

#### 2.1.1 Potential Fields

Potential Fields, first proposed by Khatib [3], with many variants implemented since, treat the robot as a point under the influence of fields generated by the goals and obstacles in the world, like an electron in an electric field. Obstacles generate repulsive forces and goals generate attractive forces, stronger near to the obstacle or goal. At every possible position, the resultant field determines the direction of motion. These methods are generally quite easy to implement, and operate extremely quickly.



**Figure 2:** An example Potential Field.

In the simplest case, we assume a simple point robot and ignore its orientation. The potential field function is  $U(q)$ , creates a force on the robot based on its position  $q = (x,y)$ . The force  $F(q)$  is based on the gradient vector of the field at that point:

$$F(q) = -\nabla U(q)$$



where

$$\nabla U(q) = \begin{bmatrix} \frac{\partial U}{\partial x} \\ \frac{\partial U}{\partial y} \end{bmatrix}$$

The resultant action on the robot is the sum of the attractive forces of the goal and the repulsive forces of obstacles:

$$F(q) = F_{att}(q) + F_{rep}(q)$$

The attractive potential of the goal can be modelled as any number of different functions. One example is a parabolic function:

$$U_{att}(q) = \frac{1}{2} k_{att} \cdot \rho_{goal}^2(q)$$

where  $\rho_{goal}(q)$  is the Euclidean distance to the goal and  $k_{att}$  is a scaling factor. The force observed by the robot (the differential of the field) will then be linear out from the goal:

$$F_{att}(q) = -\nabla U(q)$$

$$F_{att}(q) = -k_{att} \cdot \rho_{goal}(q)$$

$$F_{att}(q) = -k_{att} \cdot (q - q_{goal})$$

A repulsive potential, which should be strong close to the object and much weaker further away, might look like this:

$$U_{rep}(q) = \begin{cases} \frac{1}{2} k_{rep} \left( \frac{1}{\rho(q)} - \frac{1}{\rho_0} \right)^2 & \text{if } \rho(q) \leq \rho_0 \\ 0 & \text{if } \rho(q) \geq \rho_0 \end{cases}$$

where  $k_{rep}$  is a scaling factor,  $\rho_{goal}(q)$  is the Euclidean distance to the object and  $\rho_0$  is the maximum distance of influence of that object. This results in a force vector as follows:

$$F_{rep}(q) = -\nabla U(q)$$

$$F_{rep}(q) = \begin{cases} k_{rep} \left( \frac{1}{\rho(q)} - \frac{1}{\rho_0} \right) \frac{1}{\rho^2(q)} \frac{q - q_{obs}}{\rho(q)} & \text{if } \rho(q) \leq \rho_0 \\ 0 & \text{if } \rho(q) \geq \rho_0 \end{cases}$$

The resulting force,  $F(q) = F_{att}(q) + F_{rep}(q)$  tends towards infinity as the robot nears the obstacle so that a robot will never contact it, and will move the robot away from obstacles and towards the goal. The robot will follow the gradients of the attractive field and the sum of all the repulsive fields all the way to the goal.

However, Potential Fields has certain limitations:

- It is only a directional method, choosing robot direction as a straight line.
- It is subject to local minima, trapping the robot before the goal.
- The obstacles need to be pre-defined shapes to generate the fields.
- Oscillations are possible in the presence of multiple obstacles.
- It has difficulty navigating narrow passages.

To help overcome these limitations, the Extended Potential Field method from Khatib[4] adds a rotational potential, stronger when the robot is travelling perpendicular to the obstacle, helping to steer it away. Another recent implementation at the French LAAS-CNRS institute uses Potential Fields for lightly cluttered outdoor environments[5], building upon the Extended Potential Field method. Map grid cells are labelled as either obstacle, traversable or free. The potentials for the obstacle cells are defined as in previous methods, but for traversable cells the field is created so they are avoided where possible but traversed if necessary. This allows more flexibility for outdoor environments lacking clear distinction between obstacles and free space.

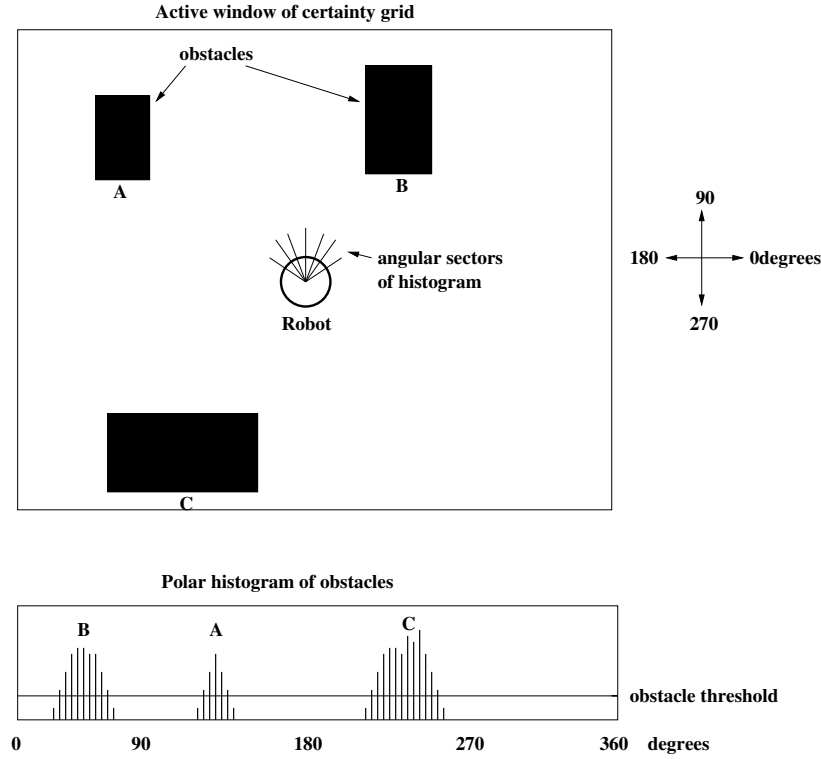
There exist many other interesting subclasses of potential fields which attempt to model the field in different ways. Circulatory fields model the obstacles with magnetic fields that the robot aligns itself[6]. It is minima free, but requires complete knowledge of the shape and location of obstacles. Another example, harmonic potential, or stream functions [7] use fluid dynamics. A source of fluid occurs at the start position of the robot, and a sink occurs at the goal, with obstacles modelled as solid objects that the fluid flows around. Once more, the robot can follow these streamlines to the goal. Both of these methods can cause chattering as the robot moves between field lines, and can be computationally expensive.

### **2.1.2 Vector Field Histogram**

Borenstein and Koren at the University of Michigan developed a variation of Potential Fields called the Vector Force Field[8]. Using a grid map, each cell generates a force on the robot scaled by a measure of certainty of it being occupied by an obstacle. This is useful because it accounts for incomplete knowledge and sensor uncertainty. However, a number of limitations to the system and Potential Fields in general were encountered. To make it function competently, it required smoothing for steering control and tricks for local minima and narrow passages.

The development of Vector Field Histogram (VFH)[9] by the same researchers aimed to overcome these limitations. One shortcoming observed with VFF and potential fields is that information is lost in reducing all data

into a simple vector. The solution was to create a polar histogram, as shown in Figure 3, accumulating the obstacles for each direction around the robot. A limited size window reduces the number of cells comprising the histogram to those near to the robot.



**Figure 3:** Vector Field Histogram.

The contents of each of the cells (i,j) in the window is represented as an obstacle vector. The direction of the vector is given by:

$$\beta_{i,j} = \arctan \frac{y_i - y_0}{x_i - x_0}$$

And its magnitude by:

$$m_{i,j} = C(i,j)^2(a - bd(i,j))$$

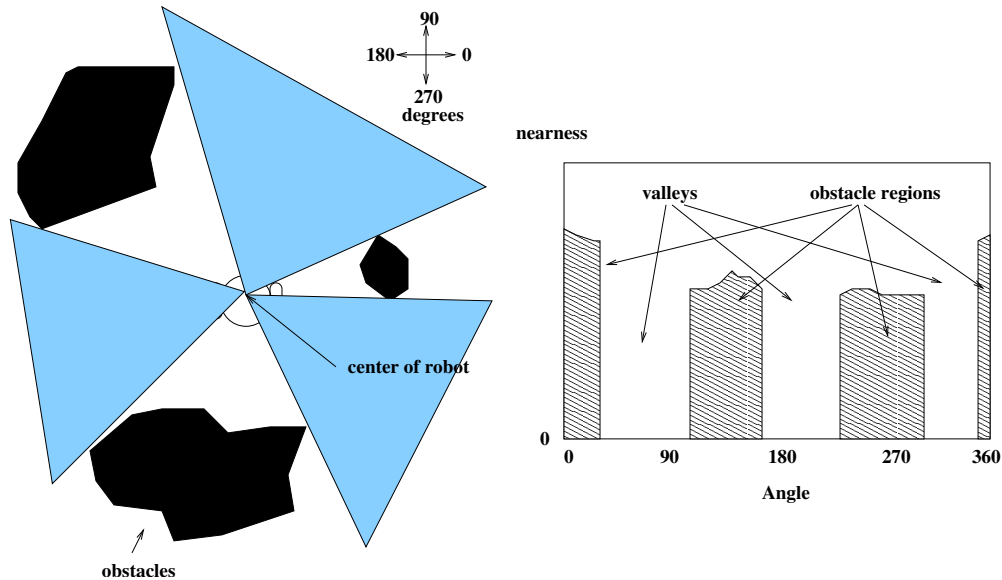
where  $C(i,j)$  is the certainty of the cell being occupied,  $a$  and  $b$  are constants and  $d(i,j)$  is the distance from the robot to the cell. The histogram is made up of a number of sectors,  $k$ , of arbitrary angle resolution. The sum of the magnitudes of these vectors which belong to each angle sector defines its value in the histogram:

$$h_k = \sum_{i,j} m_{i,j}$$

The resulting histogram will have peaks and valleys corresponding to directions with many or few obstacles, respectively. Any valley which falls below a pre-defined threshold is a candidate valley. The algorithm selects the candidate valley which most closely matches the desired target direction. Vector Field Histogram does not require command filtering and produces smoother travel than potential field methods, handling both narrow and wide openings.

### 2.1.3 Nearness Diagram

The Nearness Diagram method[10] is targeted more directly at dense and cluttered spaces. Like Vector Field Histogram, the 360 degrees around the robot are divided into sectors, with the nearness of obstacles evaluated for each sector. It creates a Point Nearness Diagram (PND) showing the nearness of obstacles to the center of the robot for each sector, as shown in Figure 4. If no obstacle is detected, the nearness is set to 0, and a valley is created. Similar to VFH, valleys between obstacles in the graph are evaluated based on width and closeness to the goal direction. The valley which best satisfies the criteria of safety and goal directedness is chosen as the selected valley.



**Figure 4:** Nearness Diagram.

The Robot Nearness Diagram (RND), an important extension to previous methods, indicates the obstacle nearness to the boundary of the robot, so the method can be applied for any number of different vehicle shapes. It uses situation based rules to determine robot behaviour, based on an estimate of safety for each of the valleys in the RND. It considers five safety situations, shown in Table 1, describing the risk of hitting an obstacle, and implements

different strategies for each. For example, in the lowest safety situation, the goal is to push the robot away from obstacles. In the second lowest, the navigator centers the robot between obstacles. In the higher safety situations, the system moves with greater goal directedness at higher speeds.

Safety Situation	Description	Action Taken
Low Safety 1	Robot chassis very close to obstacle on one side	Move away from obstacle, towards an open valley, at slow speed
Low Safety 2	Robot chassis close to obstacle on both sides	Center robot between closest obstacles, moving towards selected valley at slow speed
High Safety, Narrow Valley	Goal direction not in valley, selected valley is narrow	Robot driven in middle of the selected valley at a higher speed
High Safety, Wide Valley	Goal direction not in valley, but selected valley is wide	Robot driven along contour of the obstacles which brings it closest to the goal direction, at a higher speed
High Safety, Goal in Valley	The goal direction within the selected open valley	Robot driven directly at goal at a higher speed

*Table 1: Decision making in the Nearness Diagram method.*

#### 2.1.4 Ego-Kinematic and Ego-Dynamic Space

A further innovation these researchers introduced, related to the Dynamic Window Approach (Section 2.2.2), is the Ego-Dynamic Space[11]. It transforms the obstacle representation based on the braking constraints and sensor sampling time of the system, so that dynamics and latencies are an integral part of the obstacle representation. Therefore, the underlying obstacle avoidance technique need not concern itself with the already incorporated vehicle dynamics. Similarly, they also created an Ego-Kinematic Space[12], incorporating kinematic constraints for non-holonomic motion. Once these transformations have been completed, an obstacle avoidance technique considering only straight line motion can be used, even though the vehicle moves in circular arcs, which simplifies the problem. The Ego-Kinematic and Ego-Dynamic spaces were successfully demonstrated with both a Nearness Diagram and a potential field system.

## 2.2 Curvature Methods

Potential Fields and Vector Field Histogram evaluate straight line motion. Real robots travelling at a significant speed cannot simply change direction instantaneously. It was found more beneficial to evaluate circular arcs, accounting for some of the vehicle kinematic and dynamic constraints, such as maximum velocity, acceleration, steering angle, rate of change of steering angle, and so on. To do this, they use the space of translational and rotational velocities of the robot, and constraining the search space to

achievable velocities. This proved to be an easy extension of existing methods. For example, the VFH method described in Section 2.1.2 was extended to evaluate circular arcs in VFH+[13].

### 2.2.1 The Steering Angle Field Approach

An early approach, the Steering Angle Field approach[14] selected from a pre-determined set of discrete candidate arcs. All steering angles that would result in a collision with an obstacle are disregarded as possible commands to be sent to the vehicle, and the remaining angles are evaluated at several distance thresholds for maximizing travel and goal directedness. Velocity in this system was a secondary negotiated item, based upon the distance to obstacles.

### 2.2.2 Dynamic Window Approach

The Dynamic Window Approach by Fox[15] explicitly accounts for the finite angular and linear acceleration a robot has. Unlike the steering angle field approach, it searches the continuous space of translational and rotational velocities  $(v, \omega)$  simultaneously, rather than separate, discrete sets of steering and speed commands. Each pair of velocities results in a curvature arc given by  $c = \frac{\omega}{v}$ . The obstacles from the world are mapped into the velocity space to allow their consideration. A two-step process follows:

1) Reduce the search space from all combinations of translational and rotational velocities, to those dynamically reachable by the robot within a short, pre-determined time window, and those safe with respect to obstacles in the world. This creates the dynamic window, as shown in Figure 5. It is determined by the following formula.

#### Admissible velocities:

For a curvature  $(v, \omega)$ , which has nearest obstacle distance  $\text{dist}(v, \omega)$ , and a robot with maximum braking deceleration of  $\dot{v}_b$  and  $\dot{\omega}_b$ , then the set of admissible velocities  $V_a$  which allow the robot to stop safely is:

$$V_a = \left\{ (v, \omega) \mid v \leq \sqrt{2 \cdot \text{dist}(v, \omega) \cdot \dot{v}_b} \wedge \omega \leq \sqrt{2 \cdot \text{dist}(v, \omega) \cdot \dot{\omega}_b} \right\}$$

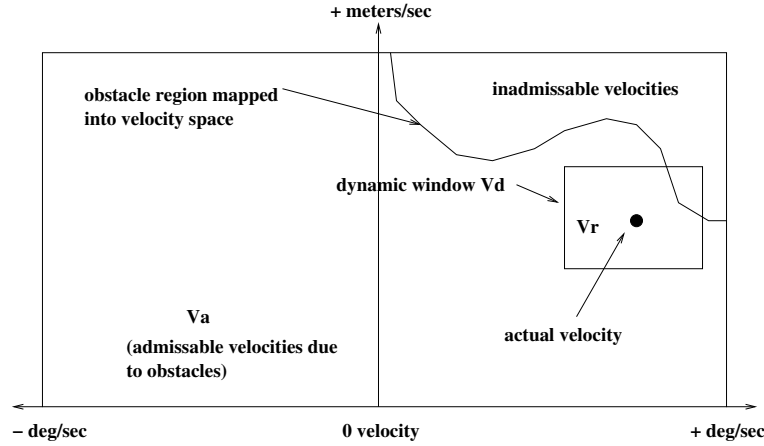
#### Dynamic window:

If acceleration or deceleration will be applied over a specified time window  $t$ , and the actual current robot velocity is  $(v_a, \omega_a)$  then the allowed velocities due to vehicle dynamics are:

$$V_d = \left\{ (v, \omega) \mid v \in [v_a - \dot{v} \cdot t, v_a + \dot{v} \cdot t] \wedge \omega \in [\omega_a - \dot{\omega} \cdot t, \omega_a + \dot{\omega} \cdot t] \right\}$$

Finally, if  $V_s$  is all possible velocities, then the set of velocities to search through is the limited set:

$$V_r = V_s \cap V_a \cap V_d$$



**Figure 5:** The Dynamic Window Approach.

2) Within this window, shown in Figure 5, find the combination of translational and rotational velocities which maximizes an objective function, accounting for goal heading, maximizing velocity, and a safe distance from obstacles. The formula for the maximum velocity set is given as:

$$G(v, \omega) = a \cdot \text{heading}(v, \omega) + b \cdot \text{velocity}(v, \omega) + c \cdot \text{dist}(v, \omega)$$

where  $a$ ,  $b$  and  $c$  are performance changing parameters, and heading is the measure of the difference between the result heading and the goal, velocity is the rate of forward progress and dist is the distance between the curvature and the nearest obstacle.

There are many follow up works to Dynamic Window, including those by Fox [16], integration with a global path planner by Brock[17], and further adaptation by Arras [18]. Schlegel [19] expanded the method to robots of any shape, and implemented on a forklift platform. Work by Ogren[20] develops a theoretical framework for the Dynamic Window Approach, to make it tractable and convergent.

### 2.2.3 Curvature-Velocity Method

The Curvature-Velocity method[21], closely related to the Dynamic Window, also evaluates curvatures in the velocity space, maximizes an objective function, and explicitly accounts for vehicle dynamics, although again in a limited fashion. Constraints on the velocity space are given by the following:

$$tv \leq tv_{max}$$

$$tv \geq -tv_{max}$$

$$rv \leq rv_{max}$$

$$rv \geq -rv_{max}$$

There are three more constraints to limit velocities achievable in the next time step ( $T_{accel}$ ), based on maximum rotational and translational accelerations ( $ra_{max}$  and  $ta_{max}$ ):

$$rv \geq rv_{cur} - (ra_{max} \times T_{accel})$$

$$rv \leq rv_{cur} + (ra_{max} \times T_{accel})$$

$$tv \leq tv_{cur} + (ta_{max} \times T_{accel})$$

Obstacles are approximated as circles, and the distance to each one along each arc is then calculated. The velocity combination is chosen by maximizing the objective function:

$$f(tv, rv) = a \cdot speed(tv) + b \cdot dist(tv, rv) + c \cdot heading(rv)$$

where:

$$speed(tv) = tv/tv_{max}$$

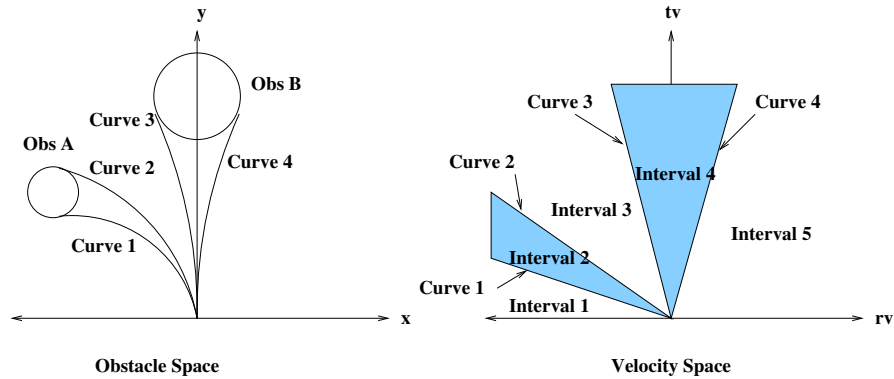
$dist(tv, rv)$  is the distance along the arc to the nearest obstacle

$$head(rv) = 1 - |\theta - rv \cdot T_{accel}| / \pi$$

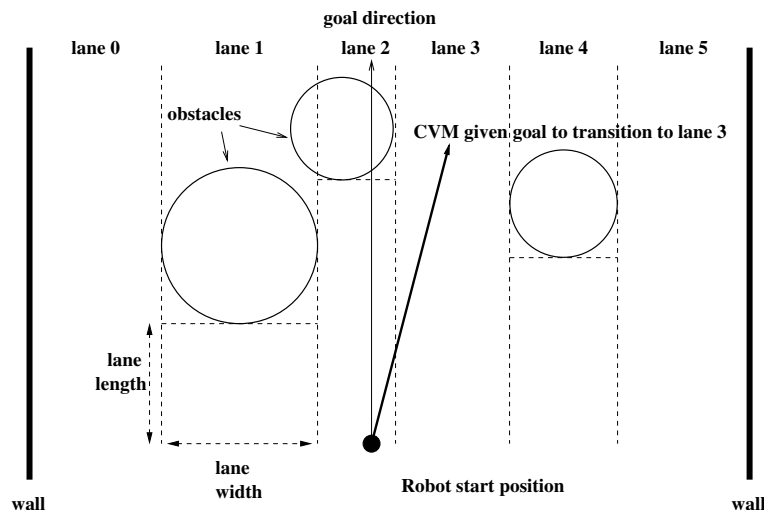
In order to achieve real-time performance, tangent curves to the obstacles are found, and the curves in between them are all assumed to have the same distance to the obstacles. The velocity space is then broken up into a number of intervals in the velocity space, as shown in Figure 6.

Based on experiences with the curvature velocity method, Ko and Simmons designed the Lane Curvature Method (LCM)[22] to improve handling of corridor intersections. Problems arose because CVM assumes that the robot travels only on fixed arcs, and the robot may change directions many times before reaching an obstacle. To address this issue, LCM models a set of driving lanes in the environment and their achievable driving distance towards the goal. The algorithm selects the best lane based on its length and width, allowing a standard CVM algorithm to make the robot move to that lane. Once in the lane, the CVM navigates until a different lane is found more advantageous. Using lanes, the robot achieves more stable behaviour over many iterations of the CVM algorithm.





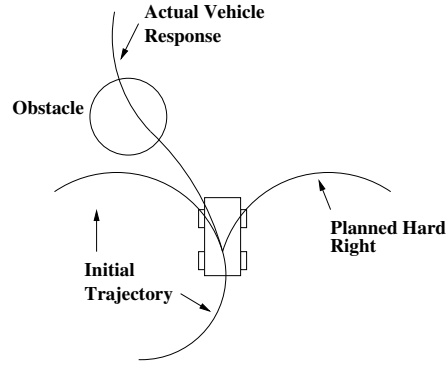
**Figure 6:** Curvature-Velocity Method.



**Figure 7:** Lane-Curvature Method.

## 2.3 Clothoid and Polynomial Methods

Some recent systems contemplate more complex vehicle motions than straight lines or circular arcs. At even moderate speeds, an arc is no longer an accurate model of vehicle response. To provide better control, curves known as clothoids are used. Figure 8 shows the difference between an arc-based and a clothoid-based navigation system. If the vehicle shown continued on its present, hard left trajectory, it would barely avoid the obstacle shown. Therefore, an intelligent control system decides to play it safe and take a hard right. However, the actual response of the vehicle, due to system steering latencies, is the curved clothoid shown in the picture, causing a collision. Because they have more real-world fidelity, clothoid systems are theoretically better than arc-based systems, but have a much vaster search space when compared with the small space of set arcs. Given growth in processing power, this approach has now become viable.



**Figure 8:** Possible problems with arc-based navigation.

A clothoid is a curve whose curvature varies linearly with length. It models an initial steering angle which then varies as the vehicle travels. A clothoid can be given by the following formula:

$$\gamma = \frac{1}{r} = c_0 + c_1 s$$

where:

$\gamma$  is the instantaneous curvature, or steering angle

$s$  is the distance travelled so far

$r$  is the radius of curvature

$c_0$  is a constant initial steering angle

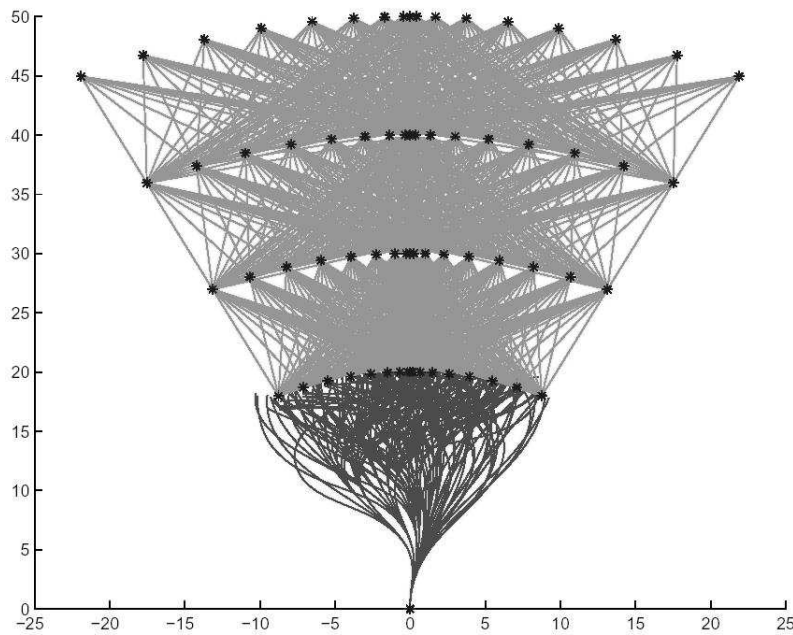
$c_1$  is a constant which forces a change in steering angle as the vehicle travels.

A clothoid which is a straight line straight ahead would have  $c_0 = 0$  and  $c_1 = 0$ . A circular arc with a fixed steering angle would have  $c_0 = k$  and  $c_1 = 0$ ,  $k$  being the constant steering angle. Finally, the more complex clothoids which more accurately describe a vehicle changing directions would have  $c_0 = k$  and  $c_1 = m$ , where  $k$  is the initial steering angle, and  $m$  is a constant which causes the steering angle to change.

### 2.3.1 DEMO II and DEMO III

The NIST DEMO II and III projects[23][24] were large, combined efforts of many groups, such as NIST, General Dynamics and Carnegie Mellon University, to demonstrate the viability of UGVs in the military reconnaissance role. It was intended for outdoor terrain, but interestingly enough, the approach taken assumes discrete obstacles. The vehicle travelled rolling grass terrain up to 35 km/hr, with large, discrete obstacles such as trees and shrubs.

The Demo II and III architectures are clothoid based, and more deliberative than other methods presented so far, with special focus on dynamic control of vehicle steering. A set of precomputed clothoid paths available given the current steering angle and velocity are used that enable the vehicle to avoid binarized obstacles. It uses offline dynamical simulations to find paths, by evaluating a database of around 15 million clothoids representing the vehicle's path over the next few seconds of driving, out to a distance of 30 meters. The picture shown in Figure 9 indicates the vehicle oriented "ego-graph", showing the pre-computed paths available to the system. The graph extends to 50 meters, the first 20 meters densely connected with smooth trajectories that are dynamically feasible, the last 30 meters connected by straight lines. Features closer to the vehicle are weighted more heavily, because they require more immediate decision and sensor information is more likely accurate. To reduce the search space, it starts by eliminating clothoids not feasible because of obstacles. It also eliminates those not dynamically feasible given initial steering conditions. Finally, the system chooses a path that follows the most benign terrain in the grid map, assigning costs to the remaining trajectories from sensed obstacles. The system starts the search at a pre-determined high vehicle speed. If no suitable trajectory is found, it lowers the choice of speed and recalculates.



**Figure 9:** Paths evaluated by Demo III obstacle avoidance.

This method is better than standard obstacle avoidance techniques because vehicle dynamics are accounted for very explicitly (i.e. the vehicle speed,

steering angle and rate of steering required to avoid an obstacle). Another interesting feature is use of a speed indexed clearance, which gives a wider berth to obstacles depending on vehicle speed.

Other complex curvature methods include those by Howard [25], Kelly's Ranger system (see Section 3.2.1), and later work by Nagy and Kelly [26, 27] which develops a theoretical framework for using more complex curves called including cubic curvature polynomials, to find solutions to complex motion planning problems quickly.

## 2.4 Obstacle Avoidance with Moving Obstacles

Moving obstacles, or the "asteroid avoidance problem", has been addressed little in the current literature, presumably because of the complications of estimating the velocity and direction of a moving obstacle given limited sensor technology. However, some different approaches have been shown, such as:

- Ignoring the problem, relying on an obstacle avoidance technique to be fast enough to dodge moving obstacles.
- Adding the time dimension to a path planning algorithm[28][29], assuming completely known trajectories of moving obstacles.
- Decomposing the problem into two sub-problems, path planning and velocity planning, first computing a path through the static obstacles and then finding the velocities along that path selected that avoid the moving obstacles[30].
- Extending an existing obstacle avoidance technique to include the velocity of obstacles, such as adding the velocity information as a further repulsive force in a potential field [31]. The examples below are of this type.

### 2.4.1 Velocity Obstacle Approach

A velocity obstacle [32] is a set of velocities which would result in a collision based not just on the obstacle's position, but its motion as well. The system estimates the avoiding velocities and colliding velocities, both translational and rotational, transforming the obstacle to the robot's velocity space. Planning then does not need to use functions of time. The system heuristically searches a tree of feasible maneuvers which avoid the velocity obstacles. Simulations show this method effectively negotiating freeway traffic, dealing with both static and moving obstacles. The velocity obstacles are generated as follows:

Figure 10a shows a robot A and an obstacle B in a 2-dimensional space. The two are then transformed into the configuration space of A by reducing A to a point  $\hat{A}$ , and by enlarging the radius of B by the radius of A, which then

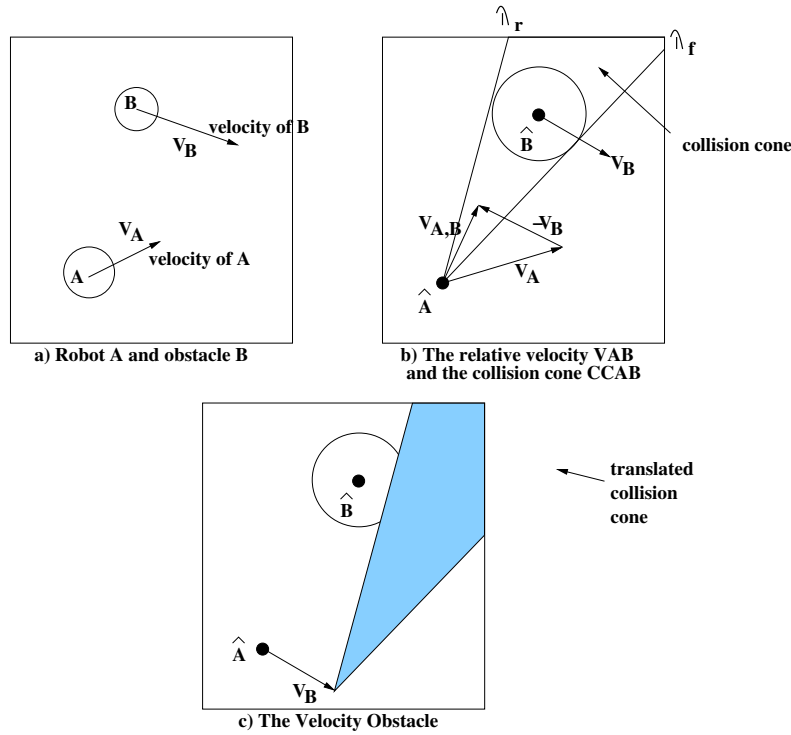
becomes  $\hat{B}$ . Each object is then shown in Figure 10b with its velocity vector attached at its center. There then exists a cone of velocities which will result in collisions between the two objects:

$$CC_{A,B} = \{V_{A,B} \mid \lambda_{A,B} \cap \hat{B} \neq \emptyset\}$$

where  $V_{A,B}$  is the relative velocity of  $\hat{A}$  with respect to  $\hat{B}$ , and  $\lambda_{A,B}$  is the line between  $\hat{A}$  and  $\hat{B}$ . In the diagram (Figure 10c) you can see that the obstacle  $\hat{B}$  is framed by the line of forward and reverse relative velocities  $\lambda_f$  and  $\lambda_r$  respectively. Any relative velocity between  $\hat{A}$  and  $\hat{B}$ , which reside in this cone will result in a collision. However, this cone only represents collisions with relative velocity between this obstacle and the robot. In order to make the cone represent absolute velocities of  $\hat{A}$ , the cone must be shifted by  $V_B$  as shown in Figure 10:

$$VO = CC_{A,B} \oplus V_B$$

This cone is then the velocity obstacle  $VO_B$ , and the planner will find a suitable trajectory around it in order to avoid a collision.



**Figure 10:** Creating a Velocity Obstacle.

Similarly, an interesting extension to the Dynamic Window approach was implemented by Castro[33]. By using two consecutive laser sensor scans, obstacles are categorized as static (unmoving) and dynamic (moving), and

their relative motion estimated. Then, the predicted collision area is modelled in the translational/rotational velocity space. Finally, the objective function from the dynamic window approach is given a fourth parameter which discourages obstacle collisions. This makes it a very simple addition to a previously demonstrated method.

#### **2.4.2 Dynamic Motion Planning Using Potential Fields**

Ge and Cui[34] extended the potential field methods with the definition of a further repulsive field based on the relative motion between the robot and its obstacle. If the obstacle is moving away, no potential is created. However, if the relative motion between the two is great, then the potential created is large.

### 3. Rough Terrain Navigation

---

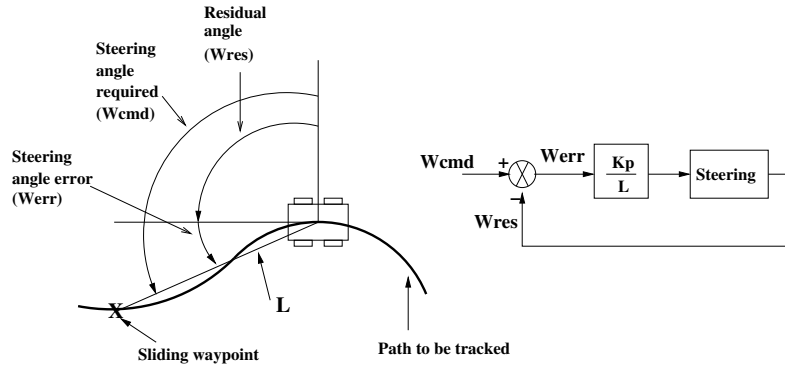
Rough terrain navigation is a much more difficult problem than indoor obstacle avoidance. In contrast to indoor obstacle methods, outdoor systems require more explicit planning mechanisms, incorporating the robot's structure, kinematics and dynamics. Mass, inertia, and vehicle stability, which aren't concerns for robots moving at slow speeds with a stable platform around discrete obstacles, now become important. Because the robot needs to be more concerned with stability, two-dimensional binary obstacle representations are also no longer effective. Therefore, most rough terrain systems use more complex models such as a 2.5D grid, or Digital Elevation Map, and model terrain roughness, slope, uncertainty, etc. Finally, rough terrain navigation is more difficult because it is harder to ensure proper execution of the planned motion.

When reviewing the methods below, it is important to understand the difference between those algorithms intended for high vehicle speed, and those for low speed. A controller for low-speed operation, such as Morphin (shown below), will use "kinematic steering" which assumes that all of the turn commands executable by the vehicle will be safe, and that the command will result in travel on a circular arc. High-speed controllers, such as the NIST Demo III architecture shown above, and systems like Ranger shown below, use "dynamic steering" which considers safety with respect to the current vehicle speed and the non-circular shape of the curve the vehicle follows. Some research for rough terrain has been aimed at military application, but a great deal has been developed to provide autonomy for the exploration of Mars.

#### 3.1 Pure Pursuit Path Tracking

One way of dealing with rough terrain navigation is to use a sufficiently detailed Global Path Planner[1], and follow the path with a feedback controller, such as the Pure Pursuit algorithm[35], or the Adaptive Pure Pursuit algorithm, optimized for rough terrain [36]. Using only Pure Pursuit makes bold assumptions that the global path planner has complete, detailed knowledge of the world, and is able to react fast enough. These assumptions rarely hold true.

Pure Pursuit was originally devised as a method to calculate the arc necessary to get a robot back onto a path. It proved to be more robust than other methods of path tracking such as the Quintic Polynomial approach or the Control Theory approach[37]. Pure pursuit calculates the curvature that will move a vehicle from its current position to some goal location. In this algorithm, the goal is a point on the planned path a fixed distance ahead of the vehicle (the lookahead distance). The algorithm pursues the moving point on the path, much the way a human drives around a curve by looking a fixed distance ahead. The general idea behind Pure Pursuit is shown in Figure 11. It is a proportional controller based on the error ( $W_{err}$ ) between the current vehicle heading and the heading needed to reach the goal ( $W_{cmd}$ ). The proportional gain  $K_p$  is normalized by the lookahead distance ( $L$ ).



**Figure 11:** Pure Pursuit Path Tracking.

The Carnegie Mellon entry in the 2004 DARPA Grand Challenge[38], Sandstorm, was able to drive a HMMV at unprecedented unmanned speeds, and follows the Pure Pursuit paradigm. Although it was equipped with sensors and some obstacle avoidance software, it required extremely detailed prior terrain data from satellite imagery, and was eventually unsuccessful at completing the Grand Challenge, although it was the first place finisher.

## 3.2 Local Navigation Using Terrain Evaluation

The approaches surveyed below in this section remove the assumption that a global path planner has complete knowledge of the world, and are reactive, sensor based approaches similar to the indoor obstacle avoidance approaches presented in Section 2. They reject the physics-based simulation (Section 3.3) and motion planning algorithms (see Section 3.5) as being too computationally expensive. Instead, the algorithms shown here rate the terrain traversability and then select a steering command which takes them across the most benign terrain.

### 3.2.1 RANGER

Ranger[36], or Real-Time Autonomous Navigator with Geometric Engine, is a product of Alonzo Kelly at Carnegie Mellon University, and made several very key contributions to navigation of UGVs. It is based on earlier Carnegie Mellon products from Singh[39] and Brummit[40]. Ranger takes range data from a laser range finder or stereo vision system and builds an elevation map (2.5D map) of the terrain in front of the robot. Then, using a high fidelity vehicle model, the system simulates possible trajectories over the terrain ahead of it. It computes the vehicle configuration (pose) at projected points in the map using the 3 axis position, the 3 orientation angles and the linear and angular speed, and then measures the hazard for each candidate steering angle. It bases this analysis on tip over, collision, roll and pitch, integrating the values along a candidate trajectory, selecting the best steering command



from this analysis. The Ranger system was able to move a HMMV at speeds of up to 15 km/hr over rough terrain, for autonomous distances up to 15km. This system was a watershed in UGV navigation. There are a number of key developments which Kelly made:

- Using a feed-forward control approach, it makes decisions via forward simulation of steering commands over the perceived terrain. This is in direct contrast to Pure Pursuit and the Robot Motion Planning methods (see Section 3.5), which globally plan a path, and then uses feedback control to follow it.
- It models the vehicle as a dynamic system, in the modern control system sense, with a multivariate differential equation, providing fidelity to the vehicles dynamic response.
- Ranger carefully models the differences between ideal and actual response of the vehicle and the software control system. It accounts for delays in software processing and latencies in vehicle acceleration, braking and change in steering angle. The system understands how the vehicle will respond to requests with respect to its current direction and velocity.

In planning motion commands, there are two basic approaches. The first plans a vehicle trajectory in the space of desired positions and velocities (state or configuration space), and then determines what steering commands will get it there using an inverse model of the vehicle. This is the traditional robot motion planning paradigm (see Section 3.5), and works well for robotic manipulators. Unfortunately, this is somewhat difficult for a dynamic system like a moving vehicle, because the inverse dynamic model is difficult to create. Planning in the configuration or state space makes it is easy to evaluate which paths are desirable, but very difficult to establish which ones are actually feasible.

The alternative method plans in the vehicle command space. In this paradigm, we start with the feasible vehicle commands, and then determine the vehicle trajectory that will result from each one. Instead of evaluating many different paths, determining the best, and trying to get a controller to follow the path, it simulates the different available vehicle commands, and finds out which one results in the best path. This method requires a forward dynamic model of the vehicle, which is much simpler than the inverse model. Any trajectories created are inherently feasible to be executed. The only catch is that a good forward model of the vehicle is necessary, or the chosen motions may not result in the trajectories estimated.

The difference between the feed-forward model (command space planning) and the inverse model (state space planning) is shown in Figure 12. This feed-forward approach is used by Ranger, its successor Morphin (Section



**Figure 12:** Inverse and forward vehicle models.

3.2.2), the Physics-Based Simulation methods (see Section 3.3.1), as well as by many of the obstacle avoidance techniques shown earlier, like Curvature-Velocity Method (Section 2.2.3), and NIST Demo III (Section 2.3.1).

The general equations for the model of a car-like vehicle, are simply the equations of dead-reckoning:

$\frac{dx(t)}{dt} = V(t)\cos\psi(t)$	$x(t) = x_0 + \int_0^t V(t)\cos(\psi(t))dt$
$\frac{dy(t)}{dt} = V(t)\sin\psi(t)$	$y(t) = y_0 + \int_0^t V(t)\sin(\psi(t))dt$
$\frac{d\psi(t)}{dt} = V(t)\kappa(t)$	$\psi(t) = \psi_0 + \int_0^t V(t)\cos(\kappa(t))dt$

where  $V(t)$  is the commanded vehicle velocity,  $\kappa(t)$  is the commanded vehicle steering angle,  $\psi(t)$  is the resultant vehicle heading and  $x(t)$  and  $y(t)$  make up the resultant vehicle position.

It can be seen that the forward model, which determines the  $(x,y)$  vehicle position and heading from the chosen steering angle,  $\kappa(t)$ , and speed,  $V(t)$ , is quite simple. On the other hand, finding a steering angle and velocity to reach a pre-selected  $x,y$  pose and heading would be much more difficult.

The linear state space model used to forward simulate the vehicle motion in Ranger can be generalized as:

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu}$$

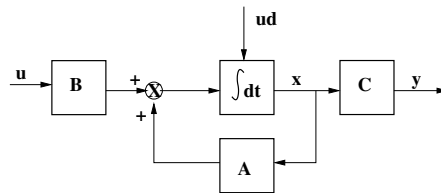
$$\mathbf{y} = \mathbf{Cx} + \mathbf{Du}$$

where:

- $\mathbf{x}$  is the state vector of the vehicle, which includes the speed and steering set points, 3D position, 3 axis orientation and linear and angular velocity.
- $\mathbf{y}$  is the output vector, or the continuous expression of vehicle hazards.
- $\mathbf{u}$  is the control vector, which includes vehicle steering, brake and speed commands.

- $u_d$  models the terrain disturbances.
- A is the “systems dynamics matrix”, which models actuator constraints, kinematics, dynamics. It propagates the vehicle forward in time.
- B is the “input distribution matrix” which transforms the control vector into its influences on state vector  $x$ .
- C is the hazard assessment step for the path being considered.

Figure 13 shows the resultant feed-forward vehicle model.



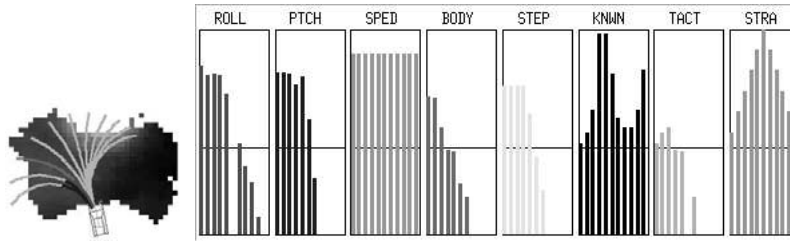
**Figure 13:** Feed-forward model in Ranger.

The hazard assessment step, or component “C” of the model shown in Figure 13, requires the assessment of the terrain for safety along the path which results from the steering command being simulated. Several different hazards are evaluated by Ranger, including:

- Tip over - True if the weight vector of the vehicle is outside the polygon formed by the wheel contact points.
- Body Collision - Collision of the underbody with the terrain.
- Discrete Obstacles - Regions of high terrain when compared to the surrounding terrain.
- Unknown terrain - Regions which the sensors cannot see, are considered hazardous.

All of the hazards are integrated over the length of the trajectory, giving an estimate for the quality of that steering command, as shown in Figure 14. The resulting tactical vote for each steering command is combined with a strategic vote to provide goal directedness. The best strategic vote which meets the minimum criteria for the tactical vote is selected.

Despite the innovations, Ranger has some limitations. Because it is “path-based”, it assumes that the vehicle will follow the intended trajectory exactly, and does not account for uncertainty in the execution of commands, the sensing, or the vehicle dynamic models used to project paths.



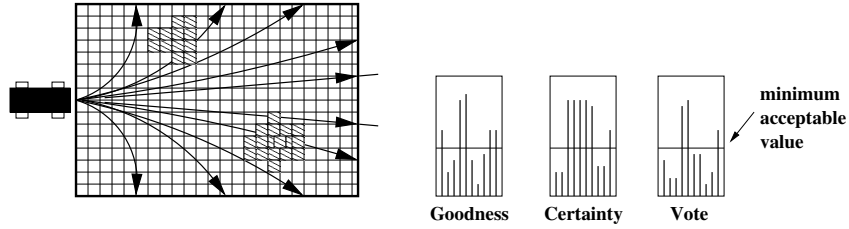
*Figure 14: Hazard assessment in Ranger.*

### 3.2.2 Morphin/Gestalt

The Morphin system[41], a direct descendant of Ranger, was intended to provide navigation for a Mars rover and, therefore, has less emphasis on high speed navigation and steering dynamics. It instead focuses on statistically rating the terrain according to roughness, slope and step hazard, based on data from stereo cameras. Whereas Ranger is a “trajectory-based” system, Morphin is “area-based”. This system works by mapping range data from the area around the robot into a grid, with cell sizes of 25cm. The grid cells are then grouped into overlapping areas, or patches, which are 1.25m on a side. By using larger “patches” of grid cells, which effectively smooth out sensing errors, it is able to better account for sensing uncertainty. The following steps are performed on the grid patches:

1. At each iteration of the algorithm, it populates the grid with the distance reading points from stereo vision.
2. Within each patch, a plane is fit to the range readings using the least squares method, and the roll and pitch of that plane is stored.
3. The maximum residual of the range points for that patch to the plane is calculated to give an estimate of roughness.
4. The roll, pitch and roughness for each patch are each normalized between 0 and 1.
5. The minimum of the roll, pitch and roughness measurements is declared to be the “goodness” for that patch, as this represents the worst possible rating for that patch.
6. A normalized certainty for each patch, also between 0 and 1, is calculated based on the number and distribution of range points, to prevent the algorithm from making decisions based on spotty information.

The map combines the overlapping portion of the goodness maps to take advantage of multiple sensor readings, and averages their findings to make decisions. Data is aged by reducing its certainty rating at every iteration, so



**Figure 15:** Arc evaluation in Morphin.

old data is used, but new data is preferred. Once the system has a “goodness” map created, it then evaluates a set of predetermined candidate arcs, as shown in Figure 15, by integrating the goodness of the patches along its length. The metric, or “goodness” of the arc, is as follows:

$$G = \frac{\int w(s)c(s)g(s)ds}{\int w(s)c(s)ds}$$

where  $c(s)$  is the certainty measure of the cells as calculated earlier,  $g(s)$  is the goodness of each patch, and  $w(s)$  is a function which discounts obstacles at the far end of each evaluated arc.

The system also calculates an arc certainty from the cell certainties:

$$C = \frac{\int w(s)c(s)ds}{\int w(s)ds}$$

Once these two metrics have been calculated, the system determines a vote for each arc by a combination of certainty and goodness, weighted for goal directedness. It will veto any paths as completely untraversable which fall below a given threshold. The candidate steering angle with the best vote is the chosen direction of travel.

A method closely related to Morphin, called Gestalt[42], from NASA’s Jet Propulsion Laboratory, added two more considerations to the goodness of each cell. The step hazard is found as the maximum elevation difference between the cell and adjacent cells. The border hazard is a set maximum goodness the cell can have if it borders a cell which is completely unknown.

Terrain analysis systems like Morphin have certain advantageous characteristics. Firstly, they are able to account for physical stability of the vehicle without complicated physical modelling. Secondly, it is easy to adapt parameters of acceptable roughness and slope depending on the robot platform and its mobility characteristics.

Systems like Morphin also have a few limitations. They require a high density of data regarding the terrain. At higher speeds, with farther look-ahead, it would be difficult to provide the necessary information from current sensing technology. Furthermore, unlike their predecessor Ranger, Morphin and Gestalt do not handle high vehicle speeds and dynamics. There is no concept of steering latency, and the resulting clothoid paths. This is quite inappropriate for UGVs operating at higher speeds. However, there is a significant window to adapt this type of statistical terrain analysis beyond application for mars rovers to high speed navigation for UGVs. In fact, predecessors to Morphin and Gestalt from Carnegie Mellon University were designed for UGV application. Smarty[43] uses statistical analysis and arc based decision making, but categorizes cells only as either traversable or non-traversable based on laser range data. Ganesha[44] does the same based on sonar sensors, rather than stereo or laser range information.

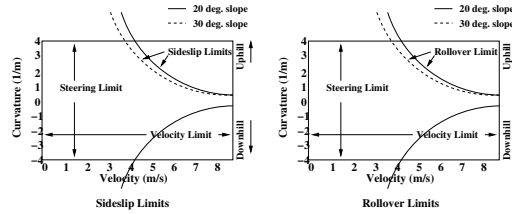
Another work which uses statistical terrain analysis of roughness and slope similar to Morphin is presented by Guo[45]. However, Guo felt that the feed-forward method of Ranger requires a system model more accurate than is practical, and is not robust to uncertainty. In contrast, Guo uses an A\* path planner to find the best trajectory over the terrain, and feedback controller to follow the trajectory. The A\* path is smoothed using interpolation, and then a set of angular and rotational velocities is assigned for each step.

### **3.2.3 High Speed Hazard Avoidance**

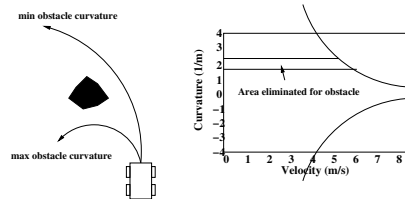
Matthew Spenko at MIT has developed a high speed rough terrain technique[46] strongly derivative of the Dynamic Window technique (Section 2.2.2). It is based on the principle that any maneuvers must take into account vehicle dynamics, vehicle/terrain interaction and performance limits of the vehicle, and relies on a library of pre-computed maneuvers to dynamically transition the vehicle from one stable motion trajectory to another. This system shows promise, but has only been shown in simulation.

The technique uses the concept of a trajectory space (the space of instantaneous curvature and velocity). This space is segmented to eliminate inadmissible trajectories based upon steering mechanism and power train limits of vehicle. Furthermore, it segments a portion of the trajectory space based on velocity and curvature combinations which would result in side-slip or roll-over, calculated using a physics-based model of the vehicle (orientation, mass, gravity and centripetal forces). The trajectory space and eliminated segments are shown in Figure 16. Finally, obstacles like trees, water holes and rocks block out a range of curvatures, as shown in Figure 17.

Other obstacles like ditches and knolls block out a range of velocities. For example, if the vehicle were to cross a ditch, it can travel at low speed, going in and out of the ditch, or it can travel at high speeds and jump over the ditch.

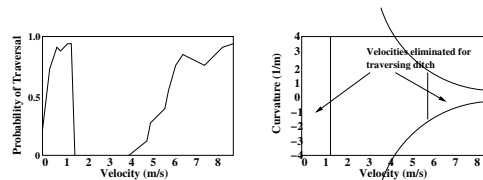


**Figure 16:** Trajectory space for high speed hazard avoidance.



**Figure 17:** Curvatures eliminated for obstacles.

The danger from impacting the far side eliminates traversing at medium speeds, as shown in Figure 18. These types of obstacles are referred to as “velocity dependant hazards”.

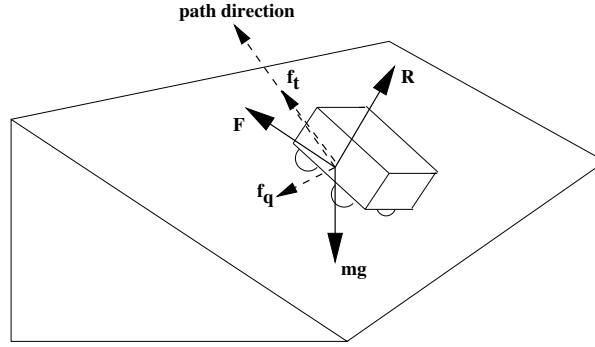


**Figure 18:** Velocities eliminated for negative obstacles.

This work by Spenko is based upon statistical motion analysis from work by Iagnemma[47] and Golda[48]. Spenko’s approach shows promise, but there are large sensing requirements to model the terrain accurately enough for this system, and it has been shown in simulation only.

### 3.3 Physics-Based Vehicle Modeling for Local Navigation

Researchers have proposed using complex vehicle models to plan the motion of robots over rough terrain. They have added much richer representations of the terrain using constructs like cubic splines, rather than the simple 2.5D elevation map used by methods shown so far, and simulate the vehicle motion over the terrain. Unfortunately, due to the complexity of physics based approaches, real-time operation necessary for fast moving vehicles is unfeasible, and are instead applied to slow moving Mars rovers.



**Figure 19:** Forces used in Shiller's rover navigator.

### 3.3.1 Physics Based Path Evaluation

An approach developed for Mars rovers, by Zvi Shiller[49][50], proposes that even a very simple model of the rover can provide a lot of information to the planner. In his system, a given path is evaluated for its traversability by computing the maximum speed along the path at which the vehicle is dynamically stable. The faster that a robot can safely travel along a given path, the better quality rating it is given.

Terrain is represented as smooth cubic B patch (a mesh of cubic splines). The vehicle model is a point mass, suspended above the ground at a height equal to the vehicles center of mass. It also includes the forces of friction, gravity, and normal ground force between the robot model and the terrain model. The equations for those forces used are given as follows:

$$f_t = mgk_t + m\ddot{s}$$

$$f_q = mgk_q + m\kappa n_q \dot{s}^2$$

$$R = mgk_r + m\kappa n_r \dot{s}^2$$

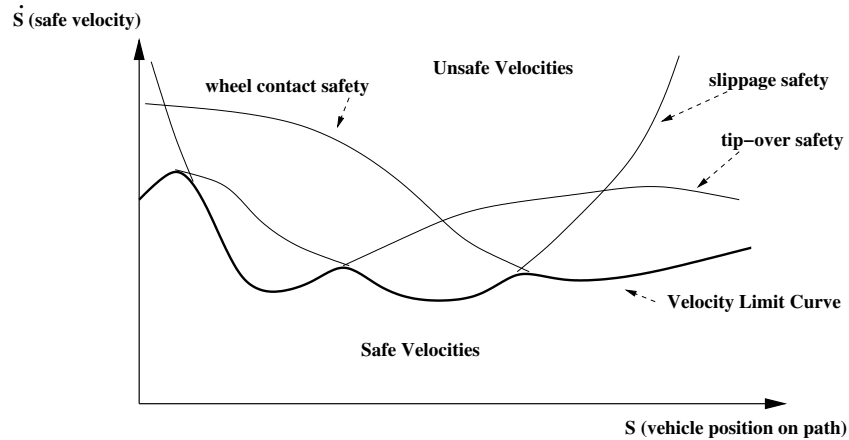
where  $F$  is the friction force,  $f_t$  and  $f_q$  are the components of the  $F$  tangent and normal to the path,  $R$  is the normal force,  $-mgk$  is the force of gravity, with  $k$  representing terrain slope.

$$\kappa = \frac{1}{\text{path curvature}}$$

while  $n$  is a vector pointing in the path direction,  $\dot{s}$  and  $\ddot{s}$  are the tangential speed and acceleration of the robot.

Path safety is then evaluated using the forces from the above equations, according to three constraints:





**Figure 20:** Shiller's velocity limit curves.

1. Sliding between the rover and the terrain, based on the normal force and friction.
2. Wheel contact between the rover and ground, by ensuring the normal force on the vehicle is always positive.
3. Tip-over constraint evaluated using the distance between wheels and height of center of gravity based on the normal and friction forces.

The robot's velocity limit (maximum safe velocity) along a path is calculated for each of the above constraints. The combined constraints for an example path can be seen in Figure 20. The lowest velocity limit from each of the three constraints is used to rate that path. As a final step, a cost function finds the optimal path based on the distance to the goal divided by the lowest velocity limits previously calculated.

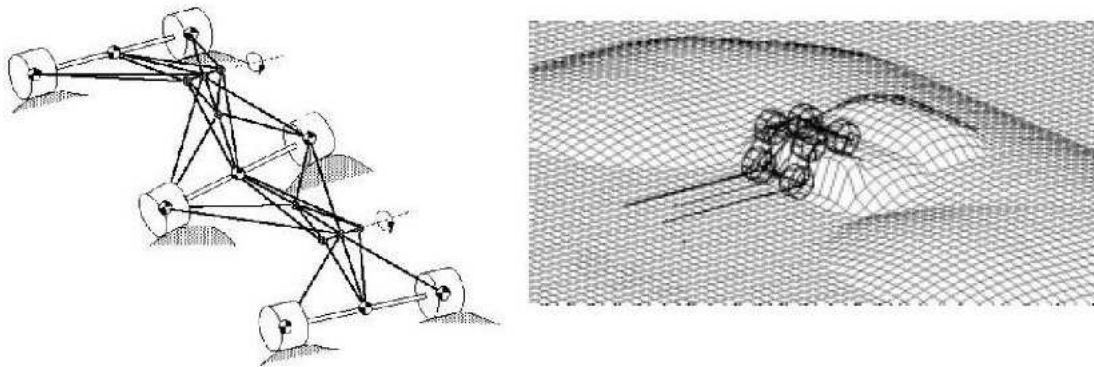
This method is computationally complex, creating a mesh for the terrain and simulating many robot motions over that mesh. However, given adequately sensed terrain, it would provide more fidelity to the robot's actual performance, but has only been shown in simulation.

### 3.3.2 Physical Modeling with Dynamic and Contact Constraints

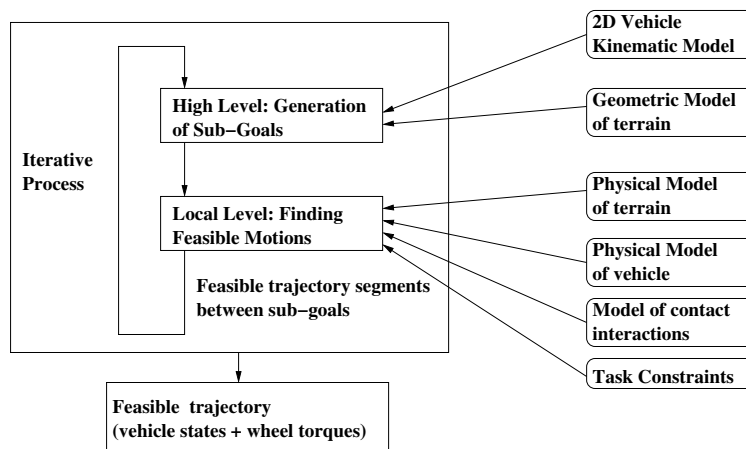
Physics-based simulation was extended by Moez Cherif [51][52][53], again intended for Mars rover control, using a more complex vehicle model. It considers articulation of the axles of the vehicle chassis and the dynamics of the robot interacting with terrain, estimating characteristics of friction and deformability. Rigid body dynamics represent the controlled parts of the vehicle, such as the driving wheels and steering, while compliant, discrete

physical structures model the passive joint mechanisms. In simulation this approach successfully captures the behaviour of the robot.

Cherif uses a two level planner: the top level (global) planner creates a set of sub-goals for the lower level (local level) planner. The global planner considers only the 3D information for the problem, using an A\* type search with a kinematic model of the vehicle, and a 3D geometric model of the terrain. The local planner works in the entire state space of the vehicle. The local planner considers distribution of soil contact, minimizing slippage at the wheels, satisfying velocity bounds, and available torque and accelerations. It also uses a complete physical model of the vehicle, a better physical model of the terrain, and a model of the soil contact interactions. The vehicle and terrain models are shown in Figure 21. As shown in Figure 22, the two planning levels are iteratively interleaved until a solution is found, unique among current research.



**Figure 21:** Physical models used by Cherif.



**Figure 22:** Iterative planning with physics-based modeling.

A couple of limitations make Cherif's work inapplicable to UGVs. Because of the complex vehicle model, it is unable to attain real-time performance. This is acceptable for a Mars rover, but not for a vehicle travelling at higher speeds. Secondly, it requires a very complete and accurate terrain model, including soil deformation and friction, not practical given current sensing technology.

Another physics based method is presented by Karl Iagnemma. [54], again using A\* graph search and a physics based model. It relaxes the requirements for complete knowledge of the terrain, analyzing roughness and accounting for uncertainty as well. Static model based safety evaluation is studied assuming the robot moves slowly and dynamic effects are negligible, making this type of analysis insufficient for high speed.

Further physics based work for rover style vehicles is presented in a series of papers from LAAS-CNRS researchers working in France[55][56]. The paths found using a search algorithm are once again safe to a static vehicle, but ignore dynamic motion. Their method was further expanded to include partially unknown environments and uncertainty in sensing[57]. Eventually, this approach was abandoned in favour of a simpler candidate arc method[58], similar to Morphin.

### **3.4 Fuzzy Logic and Neural Networks**

Fuzzy logic is a tool that can be used for modelling the relationship between system inputs and outputs. It provides the best possible guess at the appropriate value of an output from a combination of inputs, each weighted for their importance. It uses simple if-then rules, coupled with easily understandable linguistic values, to more closely emulate the vagueness in human reasoning, useful for its robustness to noise and variation in system parameters, and ability to make decisions on imprecise and incomplete information. In addition, in strong contrast to all of the other methods for robot navigation, no explicit trajectory planning is required.

An important implementation of fuzzy logic in robot navigation is that of Seraji and Howard for a Mars rover. Their method was introduced in [59], and the system described below is found in references [60] and [61]. Their system uses "area-based" evaluation of traversability of the terrain in front of the rover. This is done by a rule-based Fuzzy Traversability Index. In this index, terrain has grades of characteristics, not just simple 0 and 1 membership. This index is simple and linguistic based, measuring areas in degrees of flat/sloped/steep, or smooth/rough/rocky, in strong contrast to analytical methods such as Morphin (see Section 3.2.2), which rely on accurate interpretation and precise definition of a traversability function.

One interesting facet is that there is no range based analysis from stereo or laser cameras, and no mapping structures, the basis for almost all other navigation algorithms. Here, the analysis is entirely based upon video images, focusing on

perception of the areas around the robot, rather than measurements of its geometry. Combining this type of perception with fuzzy logic enables a more human-like method of vehicle navigation.

In order to make it more clear how the fuzzy logic is implemented, the navigation system will be described. The first step is a terrain assessment based on roughness, slope, discontinuity and hardness.

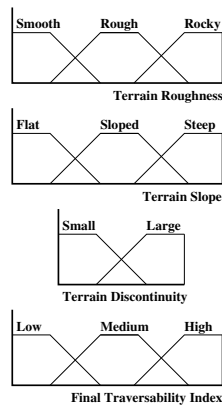
1. Roughness - Rocks in the camera image are identified by first finding the average and standard deviation of the picture color, assuming that the majority of the image right in front of the rover is non-obstacles. Then, the image is analyzed region by region. Those regions which have picture coloration differing from the rest of the picture by a certain threshold are considered rocks. This information is converted into the fuzzy sets for rock size: {SMALL, LARGE} and {FEW, MANY}. If S is rock size and C is concentration, the roughness characterization, B is as follows:
  - (a) If S is small and C is few, then B is smooth
  - (b) If S is small and C is many, then B is rough
  - (c) If S is large and C is few, then B is rough
  - (d) If S is large and C is many, then B is rocky
2. Slope - The system uses a neural network to derive slope from a pair of stereo cameras to assign descriptions to portions of the terrain as flat, sloped or steep.
3. Discontinuity - A horizon line extraction program is used to determine if there are multiple horizon lines in the image, estimating them as large or small.
4. Hardness - In order to provide for rover traction, it recognizes soft soils such as sand and gravel using a neural network and assigns a rating of soft, moderate or hard, with preference given to hard terrain.

The overall terrain traversal is classified by combining the assessments for roughness, slope, discontinuity and hardness using a set of pre-defined rules, shown in Figure 23.

A fuzzy logic control system uses this information. It consists of a number of separate behaviours:

1. Regional Traverse Behaviour - It divides the area around the robot into sectors, as shown in Figure 24, and using the terrain traversability analysis described above, it provides directions for robot motion. The speed, turn rate and desired direction are determined by fuzzy rules regarding the traversability found for the sectors.
2. Local Obstacle Avoid - If a rock is found near to the rover, then the behaviour will indicate to drive slowly. Turn angle is selected by again sectoring the world as above, with preference for avoiding nearby rocks.

Slope	Roughness	Discont.	Hardness	Trav. Index
Flat	Smooth	Small	Not Hard	Med
Flat	Smooth	Small	Hard	High
Flat	Rough	Small	Hard	Med
Flat	Rough	Small	Not Hard	Low
Sloped	Smooth	Small	Soft	Low
Sloped	Smooth	Small	Not Soft	Med
Sloped	Rough	Small	Hard	Med
Sloped	Rough	Small	Not Hard	Low
		Large		Low
	Rocky			Low
Steep				Low



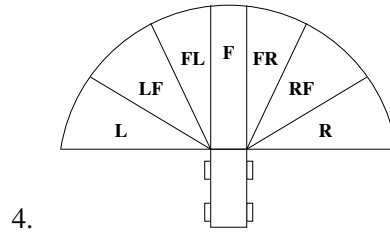
**Figure 23:** Terrain roughness fuzzy sets.

3. Global Goal Seek Behavior - This behaviour again uses fuzzy rule based logic based upon distance and angle from the desired goal direction. If the goal direction is far left, then the choice for angular velocity is far to the left, and so on.

The outputs from these behaviours, which will be desired velocities and turn rates, are then combined using weights which are themselves determined by fuzzy rules, shown in Table 2 and Figure 25. For example, if the local obstacle avoid detects an object to be very close, then its priority is increased to the system.

There are a number of good attributes to a fuzzy logic/terrain perception based approach:

- 1) Linguistic representation allows the capture of human common sense and reasoning, and makes the system easy to understand and manage.
- 2) The input variables can vary over a range without affecting the output decision.



**Figure 24:** Terrain sectors for regional traverse.

The Traverse-Terrain weight ( $t^w$ ) rules are as follows:
If forward traversability is low then $t^w$ weight is high
If forward traversability is med then $t^w$ weight is normal
If forward traversability is high then $t^w$ weight is low

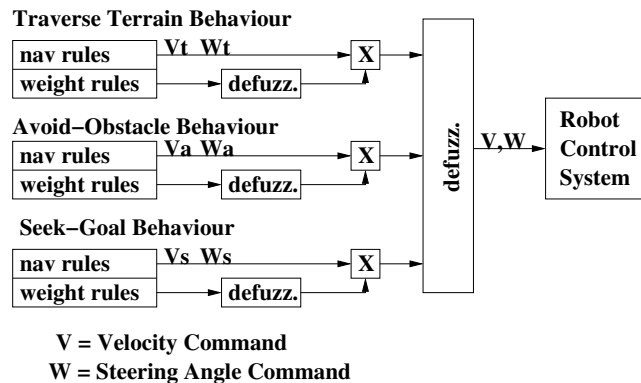
  

The Avoid Obstacle weight ( $a^w$ ) rules are as follows:
If nearest obstacle is very close then weight $a^w$ is high
If nearest obstacle is close then weight $a^w$ is nominal
If nearest obstacle is distant then weight $a^w$ is low

**Table 2:** Weight rules for fuzzy logic system.

- 3) Simplicity is inherent as each behaviour uses only a few rules and a few inputs and outputs.
- 4) Extensibility in adding rules to each individual behaviour, or behaviours to the system.
- 5) Efficiency, as evaluations are made on simple expressions.

Despite all these characteristics, the applicability of fuzzy logic to high speed UGV operation has yet to be demonstrated.



**Figure 25:** Fuzzy logic system structure.

The implementation of fuzzy logic on a Mars rover can also be found in work by Martin-Alvarez[62], which is similar to this one except that grid maps based upon stereo vision are used for terrain classification, instead of direct video perception. For some more information regarding the application of fuzzy logic to robot navigation see a survey paper by Saffiotti[63], and works by Zavlangas[64] and Tsourveloudis[65].

## **3.5 Robot Motion Planning**

In the previous sections, the problem of local navigation for UGVs was approached from a pragmatic approach, developing the navigation method entirely around the vehicle requirements. A completely different perspective, robot motion planning, has its roots in finding collision free motions for robot manipulators. Motion planners move robots between specific required poses (configurations), controlling each degree of freedom very deliberately. This makes these methods much more applicable for robot control where careful deliberation on the appropriate path is required, such as in tight locations. These methods use feedback control to follow a planned path.

### **3.5.1 Configuration and State Space**

For robot motion planning, both the vehicle and the world must be represented in some manner to evaluate plans in a search space. For many degrees of freedom, a construction called a configuration space, reduces the complexity. At every point in time, a robot has exactly one combination of its 3 axis position, 3-axis orientation, etc., called a configuration. All of the possible configurations make up the configuration space (c-space). A c-space represents each possible configuration as a single point. All of the physical obstacles from the robot's world are also mapped or transformed into the c-space. This transforms the problem from planning complex object motion to planning the motion of a point. State space is a similar construct to the c-space, which also includes parameters such as vehicle linear and angular velocity. If we consider a configuration space with  $n$  dimensions (for a robot with  $n$  degrees of freedom), its corresponding state space will have  $2n$  dimensions if we include all of the derivatives of the c-space parameters (for example,  $x$  position and its derivative  $x$  velocity).

### **3.5.2 Holonomic and Non-holonomic Constraints**

There are generally two classes of robot motion planning. Some robots can control all of the degrees of freedom independently, such as a fully actuated manipulator arm. For holonomic motion planning, a planner can simply plan the robot's motion by finding a connected path in the c-space which doesn't intersect the obstacles, using a global path planning algorithm such as A\*.

The second type plans for robots in which the degrees of freedom are not independent, and there are kinematic constraints on the motions which can be

made. For non-holonomic motion planning, a free path in the configuration space does not necessarily correspond to a feasible path, and the motion planning problem becomes much more difficult. The prime example would be a car-like vehicle, for which linear and angular position and velocity are a complicated combination of forward velocity and steering angle (see the equations in Section 3.2.1). For a car-like robot, the non-holonomic constraint for not being able to move straight sideways can be written as follows:

$$\dot{x}\sin(\psi) + \dot{y}\cos(\psi) = 0$$

where  $x, y$  are the change in vehicles position and  $\psi$  is the heading.

The difficulty arises in that, to move from one adjacent spot to another, a trajectory of arbitrary complexity may be required (think of parallel parking a car). Other types of constraints often considered in robot motion planning include kinematic constraints, such as maximum steering angle, and dynamic constraints, such as maximum velocity and maximum acceleration. These will generally appear as obstacles in the configuration space.

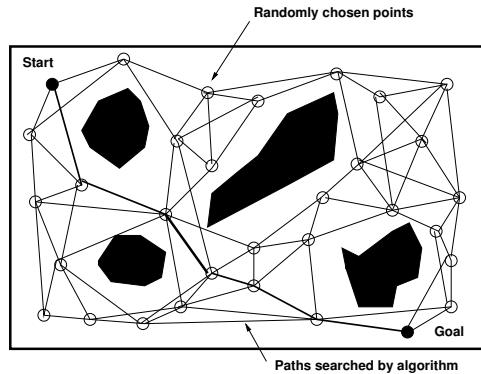
Originally, motion planners used a global path planning algorithm to search through the configuration space for a trajectory, and then used a variety of techniques to account for non-holonomic constraints. Surveys of this approach can be found in works by Latombe[66][67] and Laumond[68]. Recently Alonzo Kelly has applied the D\* global path planning algorithm to a lattice of poses to produce a motion planner[69].

### 3.5.3 Probabilistic Planning

Standard motion planners are hampered by the “curse of dimensionality”. The more degrees of freedom a robot has, the more dimensions the configuration space has, increasing the search area, and the search time goes up exponentially. A complete planner, one which is guaranteed to find the shortest path, may be unreasonably time consuming, limiting maximum vehicle speed. Probabilistic Roadmaps (PRMs) reduce this search time by sub-sampling the configuration space, and planning in this sub-set. The closer the number of sub-sampled configurations is to the total number, the more probable that the planner will find the guaranteed shortest path. PRMs are a technique for generating open-loop trajectories for nonlinear systems with state constraints.

PRMs generally work in two steps. In the first road-map phase, the planner builds an incremental road map, randomly choosing obstacle free configurations from the c-space, and linking them with admissible paths. In the second step, or query phase, paths are found between the start and goal configurations using a generic search algorithm, such as A\*.





**Figure 26:** A simplified probabilistic roadmap planner.

There are two main tenets of probabilistic planning:

- 1) Checking sampled points and connections between sampled points can be done efficiently.
- 2) A relatively small number of milestones and local paths are required to capture the connectivity of free space. A simple diagram of a probabilistic roadmap is shown in Figure 26. It is important to keep in mind that this is a 2D simplification of the high-dimensional configuration space.

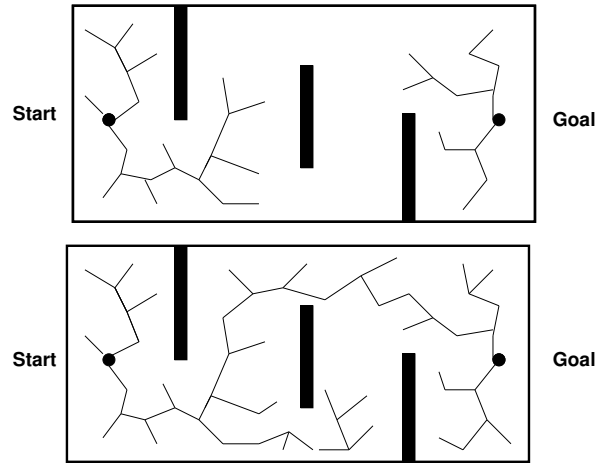
For some examples of Probabilistic Roadmap Planners, see works by Kavraki [70], Hsu[71], Leven[72].

### 3.5.4 Rapidly Exploring Random Trees

PRMs generally have difficulty in accounting for non-holonomic and dynamic constraints, since connecting the randomly selected configurations in a way which respects the constraints can be laborious. One technique plans a path in the c-space, ignoring non-holonomic constraints and then finds a controller which satisfies the robot's kinematics to follow the path[73]. This works well for manipulator robots. However, as discussed in Section 3.2.1, the inverse kinematics problem for a mobile robot is considerably more difficult for a mobile robot than for a manipulator. Manipulation control involves inverting nonlinear kinematic equations, while mobile robotics involves inverting nonlinear differential equations. A PRM may require the connection of thousands of configurations, each one akin to a nonlinear control problem, making this approach impractical. In addition, using the controller to follow the path only as closely as is feasible by the vehicle can result in executions significantly different from that planned. A final problem with standard motion planners is that the majority of them ignore dynamics entirely. For motion planning for vehicles such as helicopters and UGVs, kinodynamic

planning is crucial (as opposed to just kinematic planning), in order to account for the limits on actuator forces and response times.

A new and promising type of probabilistic planner, called the Rapidly Exploring Random Tree (RRT), shown in Figure 27, addresses these problems. Rather than randomly sampling the c-space at the start, and then trying to connect them, the planner begins at the start location and randomly expands a path, or tree, to cover the configuration or state space. Each growth of the tree is done in a way which respects the kinematic and dynamic constraints of the robot (i.e. that which can be feasibly accomplished by the vehicle). This is analogous to making a large plan as a number of randomly selected, small, feed-forward plans, inherently executable by the vehicle. Tree growth is focused to draw the expansion of the connected paths towards open areas. They rapidly search large, high dimensional spaces. The tree is grown only in ways feasible by the robot to execute, rather than by picking random points and trying to connect them in a way the robot can execute.



**Figure 27:** A simplified Rapidly Expanding Random Tree.

To grow each tree branch, non-holonomic and dynamic constraints are expressed in the form of a control system:

$$\dot{\chi} = f(\chi, \mu)$$

where:

$\chi$  is the current state in the state space. For a mobile ground robot it may be a vector consisting of  $\chi = (x, \dot{x}, y, \dot{y}, z, \dot{z}, \psi, \dot{\psi}, \theta, \dot{\theta}, \phi, \dot{\phi})$  (i.e. 3 dimensional position, roll, pitch, yaw, and all of their derivatives).

$\mu$  is the control input. For a mobile ground robot it may be a vector  $\mu = (v, \omega) \mid v \in [0, v_{max}], \omega \in [\omega_{min}, \omega_{max}]$ , where  $v$  and  $\omega$  are translational

and rotational velocities.

A general form of the RRT algorithm, taken from Lavalle[74] is as follows:

1. Two rapidly exploring trees,  $\tau_{init}$  and  $\tau_{goal}$ , are defined, one at the start and one at the goal, each containing one node,  $\chi_{init}$  and  $\chi_{goal}$  respectively.
2. A random state is selected  $\chi_{rand}$ , which is in the obstacle free portion of the state space.
3. The nearest neighbor to  $\chi_{rand}$  in the tree  $\tau_{init}$  is found, and is considered  $\chi_k$ . In the first iteration this will be  $\chi_{init}$ .
4. From the state  $\chi_k$ , all possible controls are applied to the system. Successor states to  $\chi_k$  are generated by integrating  $\dot{\chi} = f(\chi, \mu)$  for the fixed time interval  $\Delta t$ .
5. The successor to  $\chi_k$ , which is closest to  $\chi_{rand}$  and collision free is placed in the tree as  $\chi_{k+1}$ .
6. If  $\chi_{k+1}$  in  $\tau_{init}$  is within a certain distance of  $\chi_{k+1}$  in  $\tau_{goal}$  then the two trees have been connected, and our path is complete. If not, the process repeats from Step 2.

RRTs are well suited to real-time application, but cannot guarantee an optimal path, or that they will find one where one exists. However, for the local navigation problem this is of less concern. For examples, see "Randomized Kinodynamic Planning" by Lavalle[74] and Kuffner[75]. The power of these methods is evident in work by Frazzoli[76], who was able to control both a helicopter and a mobile robot with the same RRT algorithm. Although only shown in simulation, the fact that he controlled two different types of robot with strong dynamic constraints, even in the presence of moving obstacles shows great promise for the method.

### 3.5.5 Randomized Motion Planning on Rough Terrain

So far, the application of RRT methods to UGVs has been limited, but they show promise. They are becoming more practical with respect to real-time execution, but it is unclear how the consideration of terrain characteristics will be accomplished. RRTs were used by Bruce[77] to provide motion planning on real robots for a RoboCup soccer team. Biased RRTs[78] are able to consider costs of traversal in the planning algorithm. The next development will be to reconcile these concepts with rough terrain navigation in the face of sensor noise and uncertainty. Early attempts so far include works by Kobilarov[79], who modelled the terrain as a set of piecewise planar surfaces, and then used an RRT to plan motion from one to another. Kluge[80] used a biased RRT to create multiple forward paths, and then evaluated them for

nearness to obstacles, length and pitch and roll variations. The best RRT path was then selected and followed with a Pure Pursuit path tracker. See Urmson's thesis proposal [81] for a further discussion of the applicability of RRT methods to rough terrain navigation.

## 4. Conclusions and Future Work

---

As has been shown, the local navigation problem has been largely solved for indoor robots, but there is much work to be done for outdoor robots operating on rough terrain at higher speeds. There remain a number of unanswered questions, and future research directions. For example, is it better to plan a complex path and follow it with a feedback controller, relying on the fact that systems will always have detailed a priori knowledge, or to use a set of simple candidate arcs for decision making and react to the terrain as it is encountered? It remains to be seen whether a configuration space motion planning algorithm (like Rapidly Exploring Random Trees) will be more effective, or if a simpler, less deliberative feed-forward approach is better (like Ranger). It also remains to be seen whether it will be more successful to use detailed physics-based modeling to estimate vehicle/terrain interaction with future increases in processor speed (such as Cherif and Shiller propose), or to rate the terrain traversability based on simple metrics (like Morphin or Howard's Fuzzy Logic approach). At what speed and vehicle size is it necessary to account for vehicle dynamics and are clothoid methods of steering control necessary? For most cases, is it just as effective to use an arc-based method? Will complex motion planning algorithms such as RRTs be able to cope with complex terrain analysis?

Furthermore, there has been little or no progress for local navigation of UGVs in urban or 3D environments. There is little guidance available for path planning of more dexterous platforms like legged robots, or for robots like the Pakbot operating in buildings with stairs and other 3D obstacles. These issues will all need to be addressed to allow effective UGV operation in modern urban combat applications.

Obviously, different methods for local navigation have different strengths and weaknesses. They are more or less applicable for different navigation regimes. Obstacle avoidance techniques work well for slow motion where dynamics aren't a concern, but there are many discrete moving obstacles. Rough terrain navigators and fuzzy logic systems can analyze terrain well, but don't always consider vehicle dynamics that well. Physical simulators work well on Mars rovers where they have time to deliberate, but are inappropriate for high speed motion. It is an open question whether it will be possible to make a catch-all method that is good for both indoor outdoor navigation, at low and high speeds. Or will it be necessary to make a system which relies on a number of different algorithms, and switches control between them? In the meantime, it will be necessary to choose the algorithm most suited to the application.

Name	Author	Year	Type	Incomplete World Knowledge	Robot Kinemat- ics	Robot Dynamics	Robot/ Terrain Interaction	Real-Time Operation	Implemented
Potential Fields	Khatib[3]	1985	Potential Fields	No	No	No	No	Yes	Yes
Vector Field Histogram	Borenstein[9]	1991	Directional Histogram	Yes	No	No	No	Yes	Yes
Nearness Diagram	Montano[10]	2001	Directional Histogram	Yes	No	No	No	Yes	Yes
Ego-Kinematic/ Dynamic Space	Minguez[11]	2002	Directional Histogram	Yes	Yes	Partial	No	Yes	Yes
Dynamic Window	Fox[15]	1997	Curvature	Yes	Yes	Partial	No	Yes	Yes
Curvature-Velocity	Simmons[21]	1996	Curvature	Yes	Yes	Partial	No	Yes	Yes
Lane-Curvature	Ko[22]	1998	Curvature	Yes	Yes	Partial	No	Yes	Yes
Trajectory Based Ap- proach	Howard[25]	2000	Clothoid Trajec- tory	Yes	Yes	Partial	No	No	Simulaton
Velocity Obstacle Ap- proach	Fiorini[32]	1998	Moving Obstacles	Yes	Yes	Partial	No	Yes	Simulaton

**Table 3:** Comparison of obstacle avoidance techniques.

Name	Author	Year	Type	Incomplete World Knowledge	Robot Kinemat- ics	Robot Dynam- ics	Robot/Terrain Dynamics	Real-Time Operation	Vehicle Speed	Implemented
Outdoor Potential Fields	Haddad[5]	1998	Terrain Evalu- ation	Yes	Parital	No	No	Yes	Low	Yes
NIST Demo III	Coombs[24]	2000	Clothoid Tra- jectory	Yes	Yes	Yes	No	Yes	High	Yes
CMU Sandstorm	Urmson[38]	2004	Path Pursuit	No	Yes	Yes	No	Yes	Very High	Yes
Ranger	Kelly[36]	1995	Terrain Evalu- ation	Yes	Yes	Yes	No	Yes	High	Yes
Morphin	Singh[41]	2000	Terrain Evalu- ation	Yes	Yes	No	No	Yes	Low	Yes
Gestalt	Goldberg[42]	2002	Terrain Evalu- ation	Yes	Yes	No	No	Partial	Low	Yes
Physics Based Path Evaluation	Shiller[50]	2000	Physics-Based Simulation	No	Yes	Partial	Partial	No	Low	Simulation
Physical Modelling Approach	Cherif[51]	1999	Physics-Based Simulation	No	Yes	Yes	Yes	No	Low	Simulation
Physics Based Rover Planning	Iagnemma[54]	1999	Physics-Based Simulation	Partial	Yes	No	Yes	No	Low	Simulation
LAAS-CNRS	Hait[57]	1999	Terrain Evalu- ation	Partial	Yes	No	No	No	Low	Simulation
High Speed Hazard Avoidance	Spenko[46]	2004	Terrain Evalu- ation	No	Yes	Yes	Yes	Yes	High	Simulation
Rough Terrain Navigation	Guo[45]	2003	Terrain Evalu- ation	Yes	Yes	Yes	No	Yes	Low	Simulation
Fuzzy Logic Traversability Index	Serajii[61]	2002	Fuzzy Logic	Yes	No	No	No	Yes	Yes	Yes
Randomized Kino- dynamic Planning	Lavalle[74]	1999	Motion Planning	No	Yes	Yes	No	No	High	Simulation
Time Optimal Tra- jectory Planning	Kobilarov[79]	2004	Motion Planning	No	Yes	Yes	Yes	No	Low	Simulation

**Table 4:** Comparison of outdoor rough terrain techniques.

## References

---

1. Giesbrecht, J. (2004). Global Path Planning for Unmanned Ground Vehicles. (Technical Memorandum TM 2004-272). Defence R&D Canada - Suffield.
2. Lumelsky, V. and Stepanov, A. (1986). Dynamic Path Planning for a Mobile Automaton with Limited Information on the Environment. *IEEE Transactions on Automatic Control*, AC31(11).
3. Khatib, O. (1985). Real Time Obstacle Avoidance For Manipulators and Mobile Robots. In *Proceedings IEEE International Conference on Robotics and Automation*.
4. Khatib, M. and Chatila, R. (1995). An Extended Potential Field Approach for Mobile Robot Sensor-Based Motions. In *Proceedings of International Conference on Intelligent Autonomous Systems*, pp. 490–496.
5. Haddad, H., Khatib, M., Lacroix, S., and Chatila, R. (1998). Reactive Navigation in Outdoor Environments Using Potential Fields. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1232–1237.
6. Singh, L., Stephanou, H., and Wen, J. (1996). Real-Time Robot Motion Control with Circulatory Fields. In *Proceedings of the IEEE International Conference on Robotics and Automation*.
7. Kazemi, M. and Mehrandezh, M. (2004). Robotic Navigation Using Harmonic Function-based Probabilistic Roadmaps. In *Proceedings of IEEE International Conference on Robotics and Automation*.
8. Borenstein, J. and Koren, Y. (1989). Real-Time Obstacle Avoidance for Fast Mobile Robots. *IEEE Transactions on Systems, Man and Cybernetics*, 19(5), 1179–1187.
9. Borenstein, J. and Koren, Y. (1991). The Vector Field Histogram - Fast Obstacle Avoidance for Mobile Robots. *IEEE Transactions on Robotics and Automation*, 7(3), 535–539.
10. Minguez, J. and Montano, L. (2002). Robot Navigation in Very Complex, Dense and Cluttered Indoor Outdoor Environments. *Proceedings of the International Federation of Automatic Control*.
11. Minguez, J., Montano, L., and Khatib, O. (2002). Reactive Collision Avoidance for Navigation with Dynamic Constraints. *Proceedings of IEEE International Conference on Intelligent Robots and Systems*.
12. Minguez, J., Montano, L., and Santos-Victor, J. (2002). Reactive Navigation for Non-holonomic Robots Using the Ego-Kinematic Space. *Proceedings of the IEEE International Conference on Robotics and Automation*.



13. Ulrich, I. and Borenstein, J. (1998). VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots. *IEEE International Conference on Robotics and Automation*.
14. Feiten, W., Bauer, R., and Lawitzky, G. (1994). Robust Obstacle Avoidance in Unknown and Cramped Environments. In *Proceedings IEEE International Conference on Robotics and Automation*, pp. 2412–2417.
15. Fox, D., Burgard, W., and Thrun, S. (1997). The Dynamic Window Approach to Collision Avoidance. *IEEE Robotics and Automation Magazine*, 4(1), 23–33.
16. Fox, D., Burgard, W., and Thrun, S. (1998). A Hybrid Collision Avoidance Method for Mobile Robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1238–1243.
17. Brock, O. and Khatib, O. (1999). High-Speed Navigation Using the Global Dynamic Window Approach. In *Proceedings of the IEEE International Conference on Robotics and Automation*.
18. Arras, K., Persson, J., Tomatis, N., and Siegwart, R. (2002). Real-Time Obstacle Avoidance for Polygonal Robots with a Reduced Dynamic Window. In *IEEE International Conference on Robotics and Automation*, pp. 3050–3055.
19. Schlegel, C. (1998). Fast local obstacle avoidance under kinematic and dynamic constraints. *Proceedings of the IEEE International Conference on Robotics and Automation*.
20. Ogren, P. and Leonard, N. (2002). A Tractable Convergent Dynamic Window Approach to Obstacle Avoidance.
21. Simmons, R. (1996). The Curvature-Velocity Method for Local Obstacle Avoidance. *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 3375–3382.
22. Ko, Nak Yong and Simmons, Reid (1998). The lane curvature method for local obstacle avoidance. *Proceedings International Conference on Intelligent Robotics and Systems*.
23. Lacaze, A., DeClaris, N., Moscovitz, Y., and Murphy, K. (1998). Path Planning for Autonomous Vehicles Driving Over Rough Terrain. *ISIC/CIRA/ISA*.
24. Coombs, D. and Murphy, K. (2000). Driving Autonomously Offroad up to 35km/hr. In *Proceedings IEEE Intelligent Vehicles Symposium*.
25. Howard, Andrew, Blair, Alan, Walter, Dariusz, and Kazmierczak, Ed (2000). Motion control for fast mobile robots: a trajectory-based approach. *Australian Conference on Robotics and Automation*.
26. Nagy, B. and Kelly, A. (2001). Trajectory Generation for Car-Like Robots Using Cubic Curvature Polynomials. *Field and Service Robots*.

27. Kelly, A. and Nagy, B. (2002). Reactive Nonholonomic Trajectory Generation via Parametric Optimal Control. *International Journal of Robotics Research*.
28. Fujimura, K. and Samet, H. (1989). A hierarchical strategy for path planning among moving obstacles. *IEEE Transactions on Robotics and Automation*, 5(1), 61–69.
29. Conn, R. and Kam, M. (1998). Robot motion planning on N-dimensional star worlds among moving obstacles. *IEEE Transactions on Robotics and Automation*, 14(2), 320–325.
30. Fujimura, K. (1995). Time minimum routes in time dependant networks. *IEEE Transactions on Robotics and Automation*, 11(3), 343–351.
31. Ko, N. and Lee, B. (1996). Avoidability Measure in Moving Obstacle Avoidance Problem and its use for robot motion planning. *IEE International Conference on Intelligent Robots and Systems*.
32. Fiorini, P. and Shiller, Z. (1998). Motion Planning in Dynamic Environments using Velocity Obstacles. *International Journal on Robotics Research*, 17(7), 711–727.
33. Castro, D., Nunes, U., and Ruano, A. (2002). Obstacle Avoidance in Local Navigation. In *Proceedings of IEEE Mediterranean Conference on Control and Automation*.
34. Ge, S. and Cui, Y. (2002). Dynamic Motion Planning for Mobile Robots Using Potential Field Method. *Autonomous Robots*, 13, 207–222.
35. Coulter, R. Craig (1992). Implementation of the Pure Pursuit Path Tracking Algorithm. (Tech Report CMU-RI-TR-92-01). Carnegie Mellon University.
36. Kelly, A. J. (1995). Predictive Control Approach to the High-Speed Cross-Country Autonomous Navigation Problem. Ph.D. thesis. Carnegie Mellon University. Pittsburg, Pa.
37. Amidi, O. (1990). Integrated Mobile Robot Control. Master's thesis. Carnegie Mellon University.
38. Urmson, C. (2004). Navigation Regimes for Off-Road Autonomy. Ph.D. thesis. Carnegie Mellon University.
39. Singh, S. (1991). FastNav: A System for Fast Navigation. (Technical Report CMU-RI-TR-91-20). Carnegie Mellon University.
40. Brummit, B., Coulter, R., and Stentz, A. (1992). Dynamic Trajectory Planning for a Cross-Country Navigator. In *Proceedings of SPIE Symposium on Mobile Robots*.
41. Singh, S., Schwehr, K., Simmons, R., Smith, T., Stenz, A., Verma, V., and Yahja, A. (2000). Recent Progress In Local and Global Traversability For Planetary Rovers. *International Conference on Robotics and Automation*.

42. Goldberg, S., Maimone, M., and Matthies, L. (2002). Stereo Vision and Rover Navigation Software for Planetary Exploration. *IEEE Aerospace Conference Proceedings*.
43. Jr., M.B. Leahy and Saridis, G.N. (1994). A Behaviour Based System for Off Road Navigation. *IEEE Transactions on Robotics and Automation*, 10(6), 776–782.
44. Langer, D. and Thorpe, C. (1995). Range Sensor Based Outdoor Vehicle Navigation, Collision Avoidance and Parallel Parking. *Autonomous Robots*, 2(1), 147–161.
45. Guo, Y., Parker, L., Jung, D., and Dong, Z. (2003). Performance-based rough terrain navigation for nonholonomic mobile robots. *The 29th Annual Conference of the IEEE Industrial Electronics Society*.
46. Spenko, M., Iagnemma, K., and Dubowsky, S. (2004). High Speed Hazard Avoidance for Mobile Robots in Rough Terrain. In *Proceedings of SPIE Conference on Unmanned Ground Vehicle Technology*.
47. Iagnemma, K., Golda, D., Spenko, M., and Dubowsky, S. (2002). Experimental Study of High-Speed Rough-Terrain Mobile Robot Models for Reactive Behaviours. *Proceedings of the International Symposium on Experimental Robotics*.
48. Golda, D., Iagnemma, K., and Dubowsky, S. (2004). Probabilistic Modeling and Analysis of High-Speed Rough-Terrain Mobile Robots. *Proceedings of IEEE International Conference on Robotics and Automation*.
49. Shiller, Z. (1999). Motion Planning for a Mars Rover. *First Workshop on Robot Motion and Control (ROMOCO)*.
50. Shiller, Z. (2000). Online Suboptimal Obstacle Avoidance. *International Journal of Robotics Research*, 19(5), 480–497.
51. Cherif, M. (1999). Kinodynamic Motion Planning for All-Terrain Wheeled Vehicles. *Proceedings of IEEE International Conference on Robotics and Automation*.
52. Cherif, M. (1999). Motion Planning for All-Terrain Vehicles: A Physical Modelling Approach for Coping with Dynamic and Contact Interaction Constraints. *IEEE Transactions on Robotics and Automation*, 15(2).
53. Cherif, M., Cherif, M., Ibanez-Guzman, J., Laugier, C., , and Goh, T. (1999). Motion Planning for an All-Terrain Autonomous Vehicle. *International Conference on Field and Service Robotics*.
54. Iagnemma, K., Genot, F., and Dubowsky, S. (1999). Rapid Physics-Based Rough Terrain Rover Planning with Sensor and Control Uncertainty. In *Proceedings of IEE International Conference on Robotics and Automation*.

55. Simeon, T. and Dacre-Wright, B. (1993). A Practical Motion Planner for All-Terrain Mobile Robots. *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*.
56. Hait, A. and Simeon, T. (1996). Motion Planning on Rough Terrain for an Articulated Vehicle in the Presence of Uncertainties. *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*.
57. Hait, A., Simeon, T., and Taix, M. (1999). Robust Motion Planning for Rough Terrain Navigation. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*.
58. Bonnafous, D., Lacroix, S., and Simeon, T. (2001). Motion generation for a rover on rough terrains. In *International Conference on Intelligent Robotics and Systems*.
59. Seraji, H. (1999). Traversability Index: A New Concept for Planetary Rovers. *Proceedings of IEEE International Conference on Robotics and Automation*.
60. Howard, A., Seraji, H., and Tunstel, E. (2001). A Rule-Based Traversability Index for Mobile Robot Navigation. *IEEE International Conference Robotics and Automation*.
61. Seraji, H. and Howard, A. (2002). Behavior-Based Robot Navigation on Challenging Terrain: A Fuzzy Logic Approach. *IEEE Transactions on Robotics and Automation*, 18(3).
62. Martin-Alvarez, A., Volpe, R., Hayati, S., and Petras, R. (1999). Fuzzy Reactive Piloting for Continuous Driving of Long Range Autonomous Planetary Micro-Rovers. *Proceedings of the IEEE Aerospace Conference*.
63. Saffiotti, A. (1997). The Uses of Fuzzy Logic in Autonomous Robot Navigation. *Soft Computing*, 1(4), 180–197.
64. Zavlangas, P., Tzafestas, S., and Althoefer, K. (2000). Fuzzy Obstacle Avoidance and Navigation for Omnidirectional Mobile Robots. *Proceedings of ESIT*.
65. Tsourveloudis, N., Valavanis, K., and Hebert, T. (2001). Autonomous Vehicle Navigation Utilizing Electrostatic Potential Fields and Fuzzy Logic. *IEEE Transactions on Robotics and Automation*, 17(4), 490–497.
66. Latombe, J. (1991). Robot Motion Planning, Kluwer Academic Publishers.
67. Latombe, J.C. (1999). Motion Planning: A Journey of Robots, Molecules, Digital Actors, and Other Artifacts. *International Journal of Robotics Research*, 18(11), 1119–1128.
68. Laumond, J. (2004). Robot Motion Planning and Control. LAAS Report 97438.
69. Kelly, A., Amidi, O., Hoppold, M., and Herman, H. (2004). Toward Reliable Off-Road Autonomous Vehicle Operating in Challenging Environments. *International Symposium on Experimental Robotics*.

70. Kavraki, L., Svestka, P., Latombe, J., and Overmars, M. (1996). Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces. *IEEE Transactions on Robotics and Automation*, 12(4), 566–580.
71. Hsu, D., Latombe, J., and Motwani, R. (1999). Path planning in expansive configuration spaces. *International Journal of Computational Geometry and Applications*.
72. Leven, P. and Hutchinson, S. (2001). Robust, Compact Representations for Real-Time Path Planning in Changing Environments. *Proceedings IEEE/RSJ ICIRS*.
73. Svestka, P. and Overmars, M. (1995). Coordinated Motion Planning for Multiple Car-like robots. *Proceedings of the IEEE International Conference on Robotics and Automation*.
74. Lavalle, S. and Kuffner, J. (1999). Randomized Kinodynamic Planning. In *IEEE International Conference on Robotics and Automation*.
75. Kuffner, J. and Lavalle, S. (2000). RRT-Connect: An Efficient Approach to Single-Query Path Planning. In *IEEE Int'l Conference on Robotics and Automation*.
76. Frazzoli, E. (2001). Robust Hybrid Control for Autonomous Vehicle Motion Planning. Ph.D. thesis. MIT.
77. Bruce, J. and Veloso, M. (2002). Real-Time Randomized Path Planning for Robot Navigation. *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*.
78. Urmson, C. and Simmons, R. (2003). Approaches for Heuristically Biasing RRT Growth. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*.
79. Kobilarov, Marin and Sukhatme, Gaurav S. (2005). Near time-optimal constrained trajectory planning on outdoor terrain. *International Conference on Robotics and Automation*.
80. Kluge, K. and Morgenthaler, M. (2004). Multi-Horizon reactive and deliberative path planning for autonomous cross-country navigation. *Proceedings of SPIE Unmanned Ground Vehicle Technology VI*, pp. 461–472.
81. Urmson, C. (2002). Locally Randomized Kinodynamic Motion Planning for Robots in Extreme Terrain, Thesis Proposal. Ph.D. thesis. Carnegie Mellon University.



UNCLASSIFIED  
**SECURITY CLASSIFICATION OF FORM**  
**(highest classification of Title, Abstract, Keywords)**

<b>DOCUMENT CONTROL DATA</b>		
(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)		
<b>1. ORIGINATOR</b> (the name and address of the organization preparing the document. Organizations for who the document was prepared, e.g. Establishment sponsoring a contractor's report, or tasking agency, are entered in Section 8.)  Defence R&D Canada – Suffield PO Box 4000, Station Main Medicine Hat, AB T1A 8K6	<b>2. SECURITY CLASSIFICATION</b> (overall security classification of the document, including special warning terms if applicable)  Unclassified	
<b>3. TITLE</b> (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title).  Local Navigation for Unmanned Group Vehicles (U)		
<b>4. AUTHORS</b> (Last name, first name, middle initial. If military, show rank, e.g. Doe, Maj. John E.)  Giesbrecht, Jared L.		
<b>5. DATE OF PUBLICATION</b> (month and year of publication of document)  December 2005	<b>6a. NO. OF PAGES</b> (total containing information, include Annexes, Appendices, etc) <b>59</b>	<b>6b. NO. OF REFS</b> (total cited in document) <b>81</b>
<b>7. DESCRIPTIVE NOTES</b> (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)  Technical Memorandum		
<b>8. SPONSORING ACTIVITY</b> (the name of the department project office or laboratory sponsoring the research and development. Include the address.)		
<b>9a. PROJECT OR GRANT NO.</b> (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)	<b>9b. CONTRACT NO.</b> (If appropriate, the applicable number under which the document was written.)	
<b>10a. ORIGINATOR'S DOCUMENT NUMBER</b> (the official document number by which the document is identified by the originating activity. This number must be unique to this document.)  DRDC Suffield TM 2005-038	<b>10b. OTHER DOCUMENT NOs.</b> (Any other numbers which may be assigned this document either by the originator or by the sponsor.)	
<b>11. DOCUMENT AVAILABILITY</b> (any limitations on further dissemination of the document, other than those imposed by security classification)  ( x ) Unlimited distribution ( ) Distribution limited to defence departments and defence contractors; further distribution only as approved ( ) Distribution limited to defence departments and Canadian defence contractors; further distribution only as approved ( ) Distribution limited to government departments and agencies; further distribution only as approved ( ) Distribution limited to defence departments; further distribution only as approved ( ) Other (please specify):		
<b>12. DOCUMENT ANNOUNCEMENT</b> (any limitation to the bibliographic announcement of this document. This will normally corresponded to the Document Availability (11). However, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected).  Unlimited		

UNCLASSIFIED  
**SECURITY CLASSIFICATION OF FORM**

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C) or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

This document surveys existing methods of local navigation of mobile robots. It analyzes the state of the art in both indoor obstacle avoidance techniques and those for outdoor rough terrain. A variety of techniques such as Potential Fields, Vector Field Histogram, and Dynamic Window are examined in detail, in addition to outdoor systems such as Ranger, Morphin and the NIST Demo III architecture. Finally, it discusses the applicability of robot motion planning algorithms like Rapidly Exploring Random Trees to the robot navigation problem.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Unmanned Ground Vehicle, navigation, path planning, obstacle avoidance